粒子群优化算法及其工程应用

刘波　著

電子工業出版社
Publishing House of Electronics Industry
北京 • BEIJING

内 容 简 介

粒子群优化（PSO）算法是一种基于群体智能的新兴演化计算技术，广泛用于解决科学研究和工程实践中的优化问题。本书主要阐述 PSO 算法的基本理论及其在机械系统故障诊断和机械工程测试中的应用成果。全书共 5 章，第 1～3 章介绍了 PSO 算法的原理和各种改进、变体 PSO 算法的原理，第 4 章介绍了 PSO 算法在机械工程领域的应用，第 5 章介绍了 PSO 算法在其他工程领域的应用。

本书可作为机械工程、自动化等专业高年级本科生和研究生的学习参考书，也可供工程技术人员及研究人员参考使用。

图书在版编目（CIP）数据

粒子群优化算法及其工程应用 / 刘波著.—北京：电子工业出版社，2010.8
ISBN 978-7-121-11525-7

Ⅰ. ①粒… Ⅱ. ①刘… Ⅲ. ①电子计算机－算法理论 Ⅳ. ①TP301.6

中国版本图书馆 CIP 数据核字（2010）第 150847 号

责任编辑：朱清江
印　　刷：
装　　订：
出版发行：电子工业出版社
北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：720×1000　1/16　印张：8.5　字数：200 千字
印　　次：2010 年 8 月第 1 次印刷
定　　价：28.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

20 世纪 90 年代以来，受自然界生物的群体行为启发，研究者模拟自然界生物的群体行为来构造随机优化算法，产生了基于群体智能的新兴演化计算技术。典型的方法有 Dorigo 提出的蚁群算法和 Eberhart 与 Kennedy 提出的粒子群优化（Particle Swarm Optimization，PSO）算法。PSO 算法由于具有简洁、易于实现、没有太多调整参数及不需要梯度信息的特点，已广泛应用于函数优化、神经网络训练、模糊系统控制等领域。目前，国内有关 PSO 算法的书籍还比较少，本书在对 PSO 算法的理论进行介绍的基础上，结合作者近两年的研究成果，重点介绍了 PSO 算法在机械故障诊断和测试中的应用，期望能够对从事粒子群优化技术的研究人员有所帮助。本书主要阐述 PSO 算法的基本理论及其在机械故障诊断和机械工程测试中的应用成果。全书共 5 章，第 1～3 章介绍了 PSO 算法的原理和各种改进、变体 PSO 算法的原理，第 4 章介绍了 PSO 算法在机械工程领域的应用，第 5 章介绍了 PSO 算法在其他工程领域的应用。

本书可以作为计算机科学、机械工程、控制工程等高年级本科生、研究生和教师的参考用书，也可供从事群体智能优化技术研究和应用的广大研究人员使用和参考。

由于作者水平有限，书中的疏漏与不妥之处在所难免，望各位专家、学者和广大读者不吝指教。

作　者

目　录

MuLu

第 1 章　概　　论

优化理论与方法是一门应用性很强的学科，用于研究某些基于数学描述问题的最优解。美国工程院院士哈佛大学何毓琦（Yu-Chi Ho）教授指出“任何控制与决策问题本质均可以归结为优化问题”。工程中很多的实际问题在进行数学建模后，都可以抽象为一个组合优化问题。通过求解该类问题，可以为决策者提供最佳选择或最佳信息，即针对给出的实际问题，从众多的方案中选出最佳方案。优化问题最早可以追溯到古希腊时代的极值问题，如谷物的堆砌问题、等周问题等。但由于缺乏合适的计算工具和系统的理论指导，一直没有得到应有的发展。优化成为一门独立的学科是在 20 世纪 40 年代末，一方面，需要为实际生产中涌现的复杂优化问题提供快速而实用的优化算法；另一方面，包括泛函分析在内的数学理论的发展也进一步奠定了优化方法的理论基础。而计算机的出现为各种优化算法的快速实现提供了更为便捷的计算工具，这些因素促使优化逐渐成为一门应用广泛、生机勃勃的学科。

1.1　优化技术

优化是人们在工程技术、科学研究和经济管理等诸多领域中经常遇到的问题，在计算机科学、人工智能、运筹学等领域中占有非常重要的地位，是指在合理的时间范围内为一个优化问题寻找最优可行解的过程，其中优化问题的可行解之间是可以进行量化比较的。

1.1.1　优化技术介绍

具体来说，最优化问题就是在给定的约束条件下，寻找一组参数值，以使系统（或函数）的某些最优性度量得到满足，使系统（函数）的某些性能指标达到最大或最小。寻求问题最优可行解过程的第一步是要对问题进行描述和建立问题的数学模型，即用数学方程式和不等式来描述说明所求的最优化问题，其中包括目标函数和约束条件，而识别目标、确定目标函数的数学表达形式尤为关键。

1．变量的确定

变量是最优化问题或系统中待确定的某些关键元素。变量数和约束条件的多少

直接决定优化问题的规模大小，一般工程上最优设计问题属于中小规模的优化问题，而生产计划、调度问题中变量数可达几百个、几千个，属于大规模优化问题。

2．约束条件

目标函数求解时的某些限制称为约束，如变量取值范围的限定、可用资源的有限性以及所求问题的技术标准要求，此外还应满足物理系统的基本方程和性能方程。给出的约束条件越接近实际系统，则所求得的最优化问题的解也越接近于实际的最优解。约束条件可分为等式约束和不等式约束。

3．目标函数

最优化是有一定的标准或评价方法的，而目标函数就是评价优化效果的标准的数学描述，用 $f(x)$表示。最优化常指最小化和最大化两类问题。由于函数的最大化等价于其负的最小化，所以最小化和最大化问题在实际求解过程中并没有本质上的区别，本书中如不作特殊说明，一般指最小化问题。最优化问题的一般形式为

$$\begin{cases} \min f(x) \\ \text{s.t.} \quad g(x)=0 \\ \quad\ h(x)\geqslant 0 \\ \quad\ x\in \mathrm{A} \end{cases} \tag{1.1}$$

其中 x 为决策变量，A 为解的可行域，$f(x)$为目标函数，$g(x)=0$ 为等式约束，$h(x)\geqslant 0$ 为不等式约束。使目标函数 $f(x)$取得最优值的解称为最优解，记为

$$x^*\in a,\ \text{s.t.}\quad f(x^*)\leqslant f(x)\qquad \forall\, x\in a \tag{1.2}$$

对优化问题的分类有许多种，据不同的观点可以分为不同的类型，通常有如下不同分类方法。

（1）按是否有约束分类：取决于问题中有无约束，可分为有约束问题和无约束问题两种。

（2）按目标函数及约束函数特性分类；可分为线性规划、非线性规划、几何规划、整数规划和二次规划问题等。从计算的观点来说，这种分类法很有用，因为通常研究出的很多种方法都是对某一类问题有效。

（3）按计算复杂性分类：可分为 P 问题、NP 问题、NP-Hard 问题、NPC 问题等。

（4）按所包含变量确定性的性质分类：可分为确定性规划问题和随机规划问题。

（5）按目标函数与约束函数的可分离性分类：可分为不可分离问题和可分离问题。

下面介绍几个在优化问题中经常用到的概念，以帮助更好地理解后面的内容。

1．解之间的距离测度函数

设〈*A*,*f*〉是某优化问题的一个实例，定义 $\mathrm{Dist}: A\times A \to R^+$ 为计算该优化问题中的两个解之间的距离测度函数。距离测度函数的定义与优化问题决策变量的表示有很大关系，与优化算法的性能也有非常大的关系。

2．解的邻域

设〈*A*,*f*〉是某优化问题的一个实例，Dist 为解之间距离测度函数。*A* 上的一个映射 $N_e: x\in A \to N_e(x)\in 2^A$ 成为邻域映射，其中 2^A 表示 *A* 的所有子集组成的集合。也就是对任意一个 $v\in A$，集合 $N_e(v)\subseteq A$ 被称为 *v* 的邻域，$N_e: x\in N_e(v)$ 称为 *v* 的一个邻居。对于任意给定 $\varepsilon\in R^+$，N_e 的数学描述为：

$$\begin{aligned} N_e &: A \to 2^A \\ v &\to \{x\in A \mid \mathrm{Dist}(x,v)\leqslant \varepsilon\} \end{aligned} \tag{1.3}$$

邻域的构造也依赖于问题决策变量的表示，领域的结构在优化算法中起着非常重要的作用。有了邻域的定义以后，就可以定义局部、全局最优的概念了。

3．局部最优

设〈*A*,*f*〉是某优化问题的一个实例，N_e 为邻域函数。对于确定的 N_e，若 N_e 满足 $f(x^*)\leqslant f(x), \forall x\in N_e(x^*)$，则称 N_e 为在 *A* 上局部最优。

4．全局最优

设〈*A*,*f*〉是某优化问题的一个实例。若 N_e 满足 $f(x^*)\leqslant f(x), \forall x\in A$，则称 N_e 为在 *A* 上全局最优。

5．可接受解

设〈*A*,*f*〉是某优化问题的一个实例，N_e 为在 *A* 上局部最优。对于给定 $\varepsilon\in R^+$，集合 $C=\{x\in A:\mid f(x)-f(x^*)\mid \leqslant \varepsilon\}$ 被称为可接受解的集合。可接受解在优化问题中是非常重要的，因为在非常多的实例中，有限的时间内保证搜索到全局最优几乎是不可能的。在这种情况下，优化的目的往往是搜索一个满足条件的可以接受解。

1.1.2 优化算法

在某种意义上，所谓优化算法其实就是一种搜索过程和规则，它是基于某种思想和机制，通过一定的途径和规则求得满足用户要求的解的过程。优化的算法非常多，大致分类如图 1.1 所示，基本可分为两大类：精确算法和启发式算法。其中精确算法可对解空间进行彻底搜寻，得到全局最优解，如线性规划、整数规划、动态规划等算法。但这类算法所适用的优化问题很有限，对 NP-Hard、NPC 等类问题则通常使用启发式算法，又称近似算法，例如，邻域搜索算法和基于系统动态演示的算法等。

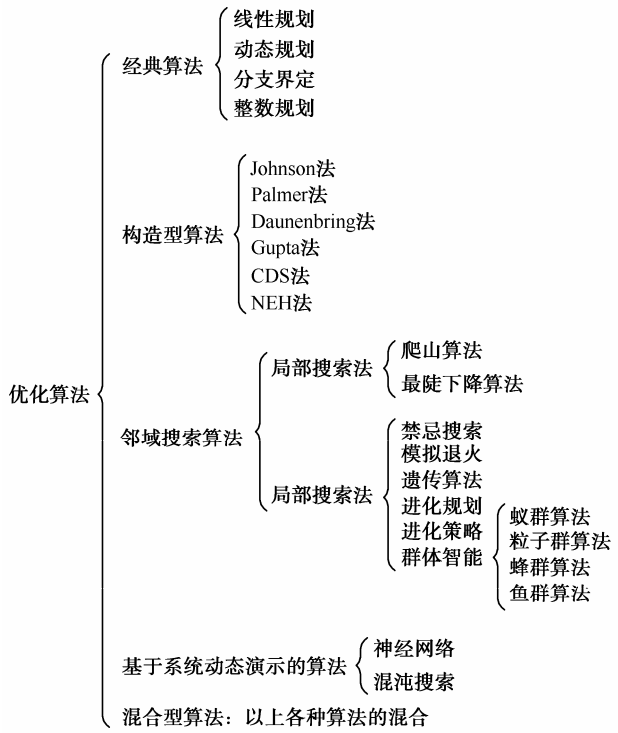

图 1.1 优化算法的组成及分类

Reeves 对启发式算法作如下定义[1]：启发式算法是一种技术，这种技术使得在可接受的计算费用内寻找好的解，但不一定能保证所得解的可行性和最优性，甚至在多数情况下，无法阐述所得解同最优解的近似程度。

根据 Reeves 的定义可知，启发式算法在多数情况不能给出最坏解的偏差程度。20 世纪 60 至 70 年代，由于人们对数学模型和最优解算法的重视，一些根据实际问题而提

出的启发式算法被称为“快速而丑陋”的算法，因此一直得不到重视。随着算法复杂性理论的完善，人们不再强调一定要得到最优解，启发式算法才开始得到广泛的重视和发展。目前优化算法的研究主要集中在启发式算法上，包括禁忌搜索、模拟退火算法、演化计算等，本书所讨论的粒子群优化（PSO）算法也属于启发式算法。当然，也有一些研究者将精确算法和启发式算法进行混合。通常，启发式优化算法给出的解是可接受解，是全局最优解的上界，因此也可被称为上界算法。也有一些算法可以获得下界，例如线性规划松弛算法、拉格朗日松弛算法等，通常被称为下界算法。

在优化理论研究领域中，一个最有趣的研究成果是 Wolpert 和 Macready 于 1997 年在《IEEE Transactions on Evolution Computation》上发表的题为“No Free Lunch Theorems for Optimization”的论文[2,3]。在这篇论文中，提出并严格论证了所谓的“无免费午餐定理”，简称 NFL 定理，NFL 定理可以描述如下：对任意给定的两种算法 A 和 B，则对于所有可能的问题集，它们的平均性能是相同的（衡量性能的指标有多种，如最优解、收敛率等）。也就是说对于所有可能的优化问题，如果 A 在某些问题类上表现比 B 好（差），那么就一定存在其他一些问题类，A 在其上的表现比 B 差（好），即 A、B 对所有问题的平均表现度量是完全一样的。

该定理表明对所有可能函数的集合而言，任何优化搜索方法的性能都是等价的，也就是说没有一个优化搜索方法是优于其他优化搜索方法的，甚至没有其他任何算法能够比搜索空间的线性列举或者纯随机搜索算法更优。那么人们就会产生这样的疑问，既然 NFL 定理表明进化算法并不比一般随机搜索算法性能好，可为什么还要对进化算法进行研究和改进呢？实际上，定理本身只是否定了去寻找一个通用的最优算法的可能性，而对于一些特定的函数集合，NFL 定理则认为存在一个适用于该集合上的优化算法，其性能在该集合上优于其他算法。在求解某些或某个特定的复杂优化问题时，可能会出现现有大多数优化方法都不适用、而适用的少数方法效果又不理想的情况，可以考虑使用进化算法。由于进化类算法本身具有很强的适用性，且对目标函数的连续性无任何要求，因此把进化算法作为求解复杂优化问题的候选算法是非常有现实意义的。

NFL 定理的主要价值在于它对研究与应用优化算法时的观念性启示作用。虽然 NFL 定理是在许多假设条件下得出的，但它仍然在很大程度上反映出了优化算法的本质。NFL 能成立的关键在于“所有可能函数的集合”，这一断言在欺骗性的和随机的函数上是成立的。当所面对的是一个大的而且形式多样的适应度值函数类时，就必须考虑算法间所表现出的 NFL 效应。由此，Christensen 等人提出了“可搜索的函数”的定义，以及适用此类函数集的一个通用的算法，可证明该算法在可搜索的函数集上的性能优于随机搜索[4]。

因此，对于所有函数集合，并不存在万能的最佳算法，所有算法在整个函数类上的平均表现度量是一样的。基于上述观点，关于优化算法的研究就应该从寻找所有可能函数类上的通用优化算法转变为以下两方面。

（1）以算法为导向，确定其适用的问题类。对于每一个算法，都有其适用和不适用的问题；对于给定的算法，尽可能通过理论分析和实际应用，找出其适用的范围，归纳特定的问题类，使其成为一个指示性的算法。

（2）以问题为导向，确定其适用的算法。对于小的特定问题类，或者一个特定的实际应用问题，设计出针对性的适用算法。实际上，大多数在优化算法方面的研究工作都是属于这一范畴的，因为它们主要是根据进化的原理设计新的算法，或者将现有算法进一步优化，以期对若干特定的函数类取得较好的优化效果。

1.2 进化计算

自 20 世纪 50 年代中期创立仿生学以来，人们从生物进化的机理中受到启发，提出了许多用于解决复杂优化问题的新方法，如遗传算法（Genetic Algorithm，GA）、进化策略 （Evolution Strategies，ES）、进化规划（Evolutionary Programming，EP）和差分进化算法（Differential Evolutionary，DE）等，这些方法被广泛应用于科学研究和实际问题求解，并取得了较好的效果。这些方法都是基于生物进化的基本思想来设计、控制和优化人工系统的，因此，这类计算方法一般统称为进化计算（Evolutionary Computation，EC）[5]。它们各具特点，分别代表了进化计算的不同侧面。

1.2.1 进化计算框架

进化算法是受生物进化论和遗传学等理论启发而形成的求解优化问题的一种随机算法，与普通的搜索算法一样属于迭代算法，即从给定的初始解中通过不断地迭代，逐步改进收敛到最优解。在进化计算中，每一次迭代被看成是一代生物个体的繁殖与进化，称为“代”（Generation）。每一个解被称为一个“个体”（Individual），用一组有序排列的基因（Gene）来表示，而每一组解被称为“人口”或“种群”（Population）。在搜索过程中则利用随机性和结构化的信息，使最满足目标的个体获得最大的生存可能，是一种概率型的算法。在自然界中，物种的性质是由染色体决定的，而染色体则是由基因有序地排列组成的。在搜索问题中，目标是由决策变量确定的，决策变量则是由一系列的分量组成的。进化算法正是人为建立并充分利用了这种相似性。

虽然进化计算的不同分支算法实现上有一些差别，但它们具有一个共同特点，即都是借助生物演化的思想和原理来解决问题。作为一种仿生的概率算法，进化算

法可以有效地对解空间进行搜索，而且算法具有较强的通用性，对问题的具体形式和领域知识依赖性不强，同时又有着本质的并行性。因此能够较快地寻找到最优解或满意解。一般而言，广义的进化算法的计算过程如下所示。

```
Begin
    首先初始化一组解。
    While（终止准则不满足）do
      根据满足目标的优劣程度来评价解的性能;
      根据所得解的性能，从当前这组解中选择一定数量的解作为进化后的解的基础;
      对所得到的解进行重组和变异等操作，结果作为进化后的解。
    End While
    输出最优解。
End
```

1.2.2　遗传算法

遗传算法（Genetic Algorithms，GA）是模拟达尔文生物进化论的自然选择和遗传学机理的生物进化过程的计算模型，是一种通过模拟自然进化过程搜索最优解的方法。遗传算法概念最早由 Bagley J.D 在 1967 年提出，是进化计算（EC）中最广为人知的一种。1975 年 Holland 出版了《Adaptation in Natural and Artificial Systems》，被认为是第一本系统论述 GA 的专著[6]。而 1989 年，Holland 的学生 Goldberg 出版的《Genetic Algorithms in Search, Optimization, and Machine Learning》则对 GA 的理论及应用作了全面系统的论述，奠定了现代遗传算法的基础[7]。

GA 的进化对象是由多个个体组成的群体，优化问题的解被表达为个体的染色体，解的值与个体的适应度值相对应，也就是个体的适应能力。群体初始化之后，先基于适应度的概率来选择父代，然后通过交叉重组和变异来维持群体的多样性，如此迭代下去，直到满足终止条件。

GA 中问题解的表示方式也就是个体的染色体，通常是固定长度的串，现在也出现了实数编码的染色体。迭代过程包含三种操作：选择、交叉重组和变异。交叉和变异都按一定的概率进行，通常被称为交叉概率 P_c 和变异概率 P_m，通常 P_c 很高而 P_m 很低。

在选择时，以适应度为选择原则。一般采用轮盘赌的方式，也就是适应度值越高的个体越有机会繁殖后代，也有一些其他选择方法，例如联赛机制等。参与被选择的个体可以是上次迭代产生的子代，也可以是上次迭代的子代和父代。根据“适者生存”原则选择下一代的个体。如式（1.4）为选中 x_i 为下一代个体的次数，显然，

适应度较高的个体，繁殖下一代的数目较多；适应度较小的个体，繁殖下一代的数目较少，甚至被淘汰。这样，就产生了对环境适应能力较强的后代。对于问题求解角度来讲，就是选择出和最优解较接近的中间解。

$$P\{\text{选中 } x_i\} = \frac{f(x_i)}{\sum_{j=1}^{n} f(x_j)} \cdot n \qquad (1.4)$$

交叉重组是由两个父代交换部分基因形成两个子代。随机地选择两个个体的相同位置，按交叉概率 P_c 在选中的位置实行交换。这个过程反映了随机信息交换，目的在于产生新的基因组合，即产生新的个体。一般采用单点等位基因交叉的方式，后来也出现了多点交叉、任意点交叉等方式。对于实数编码的染色体，则采用代数交叉，两个父代分别为 x_1 和 x_2，交叉后的子代分别为 $\lambda x_1 + (1-\lambda)x_2$ 和 $(1-\lambda)x_1 + \lambda x_2$，其中 $\lambda = \text{rand}(0,1)$。

变异是根据生物遗传中基因变异的原理，以变异概率 P_m 对某些个体的某些位执行变异。在变异时，对执行变异的串的对应位求反，即把 1 变为 0，把 0 变为 1。变异概率 P_m 与生物变异极小的情况一致，所以，P_m 的取值较小，一般取 0.01～0.2。

对于二进制编码的染色体，变异则是随机将某些位变反；实数编码染色体则采用在值上加正态噪声 $N(0,\delta)$ 的方法，也称为高斯变异，其中 δ 可随迭代的代数 t 的增长而递减，例如使用类似 $\delta = 1/(1+\sqrt{t})$ 的函数。

通过离散空间的 Markov 链可以证明，标准 GA 是无法保证全局收敛的[8,9]，但保存最佳个体（Elitism）的 GA 可以保证在时间无限条件下以概率 1 全局收敛的[10]。GA 的流程如下所示。

```
Begin
  初始化种群和参数;
  评价个体适应度（给出目标函数 f(x)，则 f(x_i) 称为个体 x_i 的适应度）。
  While（终止准则不满足）do
    选择（Selection）;
    交叉（Crossover）;
    变异（Mutation）;
    评价个体适应度。
  End While
  输出最优个体。
End
```

在上述步骤中初始化种群即创建一个随机的初始种群，个体记为 x_i（$i = 1,2\text{L},n$）。这个初始的群体也就是问题假设解的集合。将这些解比喻为染色体

或基因，该种群被称为第一代。问题的最优解将通过这些初始假设解进化而求出。通常算法的迭代终止准则选取以下两个。

（1）最优个体的适应度达到给定的阈值。

（2）算法的最大迭代次数到达（通常这时最优个体的适应度和群体适应度不再上升）。

由于 GA 是由进化论和遗传学机理而产生的搜索算法，所以在这个算法中会用到很多生物遗传学知识，下面对一些术语进行一下说明。

（1）染色体（Chromosome）。染色体又可以叫做基因型个体，一定数量的个体组成了群体（Population），群体中个体的数量叫做群体大小。

（2）基因（Gene）。基因是串中的元素，基因用于表示个体的特征。例如有一个串 $S=1011$，则其中的 1、0、1、1 这 4 个元素分别称为基因。它们的值称为等位基因。

（3）基因地点（Locus）。基因地点在算法中表示一个基因在串中的位置，也称为基因位置（Gene Position），有时也简称基因位。基因位置由串的左向右计算，例如在串 $S=1101$ 中，0 的基因位置是 3。

（4）基因特征值（Gene Feature）。在用串表示整数时，基因的特征值与二进制数的权一致，例如在串 $S=1011$ 中，基因位置 3 中的 1，它的基因特征值为 2；基因位置 1 中的 1，它的基因特征值为 8。

（5）适应度（Fitness）。各个个体对环境的适应程度叫做适应度。为了体现染色体的适应能力，引入了对问题中的每一个染色体都能进行度量的函数，叫适应度函数。这个函数是计算个体在群体中被使用的概率。

GA 是解决搜索问题的一种通用算法，对于各种通用问题都可以使用。同其他搜索算法相比，GA 还具有以下几方面的特点。

（1）GA 从问题解的串集开始搜索，而不是从单个解开始。这是 GA 与传统优化算法的极大区别。传统优化算法是从单个初始值迭代求最优解的，容易误入局部最优解。遗传算法从串集开始搜索，覆盖面大，利于全局择优。

（2）许多传统搜索算法都是单点搜索算法，容易陷入局部的最优解。GA 同时处理群体中的多个个体，即对搜索空间中的多个解进行评估，减少了陷入局部最优解的风险，同时算法本身易于实现并行化。

（3）GA 基本上不用搜索空间的知识或其他辅助信息，而仅用适应度函数值来评估个体，在此基础上进行遗传操作。适应度函数不仅不受连续可微的约束，而且其定义域可以任意设定。这一特点使得 GA 的应用范围大大扩展。

（4）GA不是采用确定性规则，而是采用概率的变迁规则来 指导其搜索方向。

（5）具有自组织、自适应和自学习性。GA利用进化过程获得的信息自行组织搜索时，硬度大的个体具有较高的生存概率，并获得更适应 环境的基因结构。

由于GA的整体搜索策略和优化搜索方法在计算时不依赖于梯度信息或其他辅助知识，而只需要影响搜索方向的目标函数和相应的适应度函数，所以GA提供了一种求解复杂系统问题的通用框架，它不依赖于问题的具体领域，对问题的种类有很强的 鲁棒性，所以广泛应用于许多科学领域。

（1）函数优化。函数优化是遗传算法的经典应用领域，也是 GA 进行性能评价的常用算例，许多人构造出了各种各样复杂形式的测试函数：连续函数和离散函数、凸函数和凹函数、低维函数和高维函数、单峰函数和多峰函数等。对于一些非线性、多模型、多目标的函数优化问题，用其他优化方法较难求解，而 GA 可以方便地得到较好的结果。

（2）组合优化。随着问题规模的增大，组合优化问题的搜索空间也急剧增大，有时在目前的计算上用枚举法很难求出最优解。对这类复杂的问题，人们已经意识到应把主要精力放在寻求满意解上，而GA是寻求这种满意解的最佳工具之一。实践证明，GA对于组合优化中的NP问题非常有效。例如GA已经在求解旅行商问题、背包问题、装箱问题、图形划分问题等方面得到成功的应用。

此外，GA也在生产调度问题、自动控制、机器人学、图像处理、人工生命、遗传编码和 机器学习等方面获得了广泛的运用。

1.2.3　进化规划

进化规划（Evolutionary Programming，EP）是由美国的 Fogel 于 20 世纪 60 年代提出来的，Fogel、Owens 和 Walsh 出版的《Artificial Intelligence Through Simulated Evolution》被认为是第一本关于 EP 的专著[11]。Fogel 在这部专著中系统阐述了进化规划的思想。他提出的方法与 GA 有许多共同之处，但不像 GA 那样注重父代与子代在遗传细节上的联系，而是把侧重点放在父代与子代表现行为的联系上。EP 仅使用变异与选择算子，而绝对不使用任何重组算子。其变异算子与进化策略的变异相类似，也是对父代个体采用基于正态分布的变异操作进行变异，生成相同数量的子代个体，即 m 个父代个体总共产生 m 个子代个体。EP 采用一种随机竞争选择方法，从父代和子代的并集中选择出 m 个个体构成下一代群体。其选择过程如下：对于由父代个体和子代个体组成的大小为 $2m$ 的临时群体中的每一个个体，从其他 $2m-1$ 个个体中随机等概率地选取出 q 个个体与其进行比较。在每次比较中，若该个体的适应度值不小于与之比较的个体的适应度值，则称该个体获得 1 次胜利。从 $2m$ 个个体中选择出获胜次数最多的 m 个个体作为下一代群体。

在 EP 中，假设问题的搜索空间是一个 n 维空间，则搜索点是一个 n 维向量：$\boldsymbol{X} \in \boldsymbol{R}^n$，进化群体的每个个体等同于这个 $\boldsymbol{X} \in \boldsymbol{R}^n$，所有的个体构成了整个群体。选择一个整数 N 作为群体的规模参数，随机生成搜索空间的 N 个初始个体作为初始群体。一般来说，这些个体的适应度值较差。EP 就是从这一初始群体出发，通过遗传操作，模拟进化过程，最后获得非常优秀的群体和个体。在 EP 中，个体适应度 $F(\boldsymbol{X})$ 是由它所对应的目标函数 $f(x)$ 通过某种比例变换而得到的。这种比例变换保证各个个体的适应度总取正值：

$$F(\boldsymbol{X}) = \delta[f(x)] \tag{1.5}$$

其中，δ 为某种比例变换函数。

在变异（Mutation）操作中，标准 EP 采用的是高斯变异算子。假设群体中某一个体 $\boldsymbol{X} = (x_1, x_2, \mathrm{L}\ , x_n)$，经过变异运算后得到一个新的个体 $\boldsymbol{X}' = (x_1', x_2', \mathrm{L}\ , x_n')$，则变异算子把个体 $\boldsymbol{X}$ 的每个分量 x_i 作用一个标准偏差得到新个体的组成元素。

$$x_i' = x_i + \delta_i N_i(0,1) \tag{1.6}$$

其中，$\delta_i = \beta_i \cdot f(x) + \gamma_i$，$i = 1, 2, \mathrm{L}\ , n$；$N_i(0,1)$ 表示对每个下标 i 都重新取值的均值为 0、方差为 1 的符合正态分布的随机变量；系数 β_i、γ_i 是特定的参数，一般取 $\beta_i = 1$，$\gamma_i = 0$。进化规划着眼于整个群体的进化，强调的是“物种的进化过程”。所以进化规划不使用交叉运算之类的个体重组算子，高斯变异算子的生物基础是强调个体的进化机制。

选择（Selection）体现了“适者生存”的自然法则。一般按照一种随机竞争的方式来进行（联赛选择方式），即在 N 个父辈个体 $P(t)$ 每个经过一次变异产生 N 个子代个体 $P'(t)$ 后，EP 利用一种随机 q 竞争选择方法从父辈和子代组成的共含有 $2N$ 个个体的个体集合 $\{P(t) \cup P'(t)\}$ 中选择 N 个个体，其中 $q \geqslant 1$ 是选择算子的参数。基本过程如下：对于每个个体 $\boldsymbol{X}_k \in \{P(t) \cup P'(t)\}$，从 $\{P(t) \cup P'(t)\}$ 中随机选取 q 个个体，比较这 q 个个体与个体 $\boldsymbol{X}_k$ 之间的适应度大小，以其中适应度比 $\boldsymbol{X}_k$ 的适应度还要差的个体的数目作为个体 $\boldsymbol{X}_k$ 的得分 $W_k (k = 1, 2, \mathrm{L}\ , 2N)$。在所有 $2N$ 个个体都经过了这种比较过程后，按得分 W_k 的大小做降序排列。选择前 N 个个体作为下一代群体 $P(t+1)$。

从上述选择算子可以知道，在进化过程中，每代群体中最好的个体在比较适应度大小时总被赋予了最大的得分，从而最好的个体能够确保被保留到下一代群体中。标准 EP（Classical Evolutionary Programming，CEP）的主要流程如下所示。

```
Begin
  产生初始群体;
```

```
  评价适应度（适应度函数是个体竞争的测度，控制个体生存的机会）。
  While（终止准则不满足）do
    变异（Mutation）;
    选择（Selection）;
    评价适应度。
  End While
  输出最优个体。
End
```

在 EP 中终止准则主要有以下几种。

（1）EP 已找到能接受的优秀个体。

（2）EP 已进化了预定的最大代数。

（3）在预定的代数内最适应个体的适应度无改进。

（4）最适应个体占群体的比例已达到规定的比例。

EP 的特点主要有以下几点。

（1）EP 不采用某种特定的编码结构，而直接对实际参变量（十进制）进行操作。

（2）EP 同时搜索解空间中的一群点，而非单点。EP 同时对空间中不同的区域采样，并构成不断进化的群体序列，或者说 EP 并行地爬多个山峰。这一特点使 EP 可以有效地防止搜索过程限于局部最优解，而具有较大的可能求得全局最优解。

（3）EP 对搜索空间没有任何特殊要求（如连续性等），对目标函数几乎无限制，不要求连续可微，既可以是显函数，也可以是映射矩阵甚至神经网络等隐函数，仅要求可用适应度函数评价个体。EP 的这一特点再一次拓宽了其应用领域。

（4）EP 采用概率变迁规则而非确定性规则来指导其搜索空间。EP 采用概率作为一种工具来启发搜索，有明确的搜索方向，比随机搜索方法有更高的搜索效率。

（5）EP 具有隐含并行性。不但使 EP 再次提高了搜索效率，而且易于采用并行机作并行高速运算，发展潜力很大。

EP 与 GA 作为进化算法，它们之间既有相同之处也有不同。相同之处有以下三点。

（1）操作一个候选解群体。

（2）限制一些候选解的被选择权

（3）确定一个选择准则得到下一代群体。

不同之处也有以下三点。

（1）个体的表示方法不同。GA 的处理对象不是参变量本身，而是参变量编码后的染色体串。因此存在基因型与表现型之间的映射问题。EP 不采用某种特定的编码结构，而直接对实际参变量（十进制）进行操作。

（2）遗传算子不同。GA 强调用交叉、翻转和变异的遗传算子应用到抽象的染色体；EP 强调父辈与子代之间的变异过程，重组只能应用到个体，而不能应用到整个群体。

（3）选择准则不同。在 GA 中每个父辈均有机会产生后代，而且产生后代的数量与其适应度值有关；在 EP 中每个父辈只产生一个后代，并且还要和后代一起竞争其生存权。

1.2.4　进化策略

虽然进化策略（Evolutionary Strategies，ES）的思想与 EP 的思想有很多相似之处，但它是在欧洲独立于 GA 和 EP 而发展起来的。1963 年，当时德国柏林工业大学的 ReehenbergL 和 Schwfel 等在进行风洞实验时，由于在设计中描述物体形状的参数难以用传统的方法进行优化，从而他们利用自然突变和自然选择的生物进化思想，对物体的外形参数进行随机变化并获得了较好的结果。随后，他们便对这种方法进行了深入的研究和发展，形成了进化计算的另一个分支，即进化策略（ES）[12]。早期 ES 的种群中只含有一个个体，且仅使用变异一种算子。在每一次进化过程中，通过变异后的子体与父体的比较来选择两者中的最优。这一策略称之为（1+1）策略。由于（1+l）策略存在效率较低，有时收敛不到全局最优解等不足，所以需要进行改进，其改进的方法就是增加种群内个体的数量，即后来的（μ+1）策略。此时种群内包含μ个个体，随机抽取其中的一个个体进行变异，然后更新群中的最差个体。为了进一步提高搜索的效率，随后又相继发展了（$\mu+\lambda$）策略和（μ，λ）策略。（$\mu+\lambda$）策略是根据种群内的μ个个体通过变异和重组操作产生 n 个个体，然后通过$\mu+\lambda$个个体的比较，从中选取μ个最优者；而（μ，λ）策略则是在新产生的λ（μ）个个体中通过比较选取μ个最优者。下面简单介绍 ES 的一般形式。

1．表示法和适应度值度量

在 ES 中，搜索点是 n 维向量 $x \in \boldsymbol{R}^n$，个体的适应度值等于其函数值 $\phi(a)=f(x)$。其中 x 是个体 a 的目标变量部分。此外，每个个体可以包括最多 n 个不同的方差 $c_n=\delta_i^2, i\in\{1,2,\mathrm{L}\ ,n\}$ 和最多 $n\cdot(n-1)/2$ 个协方差 $c_{ij}=\delta_{ij}^2$，$i\in\{1,2,\mathrm{L}\ ,n-1\}$，$j\in\{i+1,\mathrm{L}\ ,n\}$。从而最多有 $w=n\cdot(n-1)/2$ 个策略参数可以和目标变量组合在一起构成一个个体 $a\in \boldsymbol{I}=\boldsymbol{R}^{n+w}$。不过，一般只考虑方差，从而 $a\in \boldsymbol{I}=\boldsymbol{R}^{2n}$，有时甚至对所有目标变量只用一个共同的方差，这时 $a\in \boldsymbol{I}=\boldsymbol{R}^{n+1}$。

2．变异

进化策略中，全体 $a=(x,\delta)$ 在变异算子作用下变为 $a'=(x',\delta')$，这里

$$\begin{aligned}\delta_i' &= \delta_i \cdot \mathrm{e}^{\tau' \cdot N(0,1)+\tau \cdot N_i(0,1)} \\ x_i' &= x_i + N(0,\delta_i'),\quad i=1,2,\mathrm{L}\ ,n\end{aligned} \tag{1.7}$$

其中，$N(0,1)$ 表示具有期望值为 0、标准偏差为 1 的正态分布随机变量，τ' 和 τ 是算子集参数，分别定义整体和个体步长。

3．重组

在进化策略中，重组算子可以按下列方式产生一个个体。

无重组：

$$x_t' = x_{S,t}$$

直接重组：

$$x_t' = x_{S,t} \quad 或 \quad x_{T,t}$$

加权平均重组：

$$x_t' = x_{S,t} + k(x_{T,i} - x_{S,t})$$

下标“S”和“T”指从 $P(t)$中随机选取的两个父辈个体，$k \in [0,1]$ 为一致随机变量。

4．选择

在进化策略中，选择是按完全确定的方式进行。$(\mu,\lambda)-ES$ 是从几个子代个体集中选择 $\mu(1 \leqslant \mu \leqslant \lambda)$个最好的个体；$(\mu+\lambda)-ES$ 是从父辈和子代个体的并集中选择μ 个最好的个体。虽然 $(\mu+\lambda)-ES$ 保留最优的个体能保证性能单调提高，但这种策略不能处理变化的环境，因此，目前选用最多的还是 $(\mu,\lambda)-ES$，研究表明比率 $\mu/\lambda \approx 1/T$ 是最优的。

由上面的讨论可知，进化策略算法的流程如下所示。

```
Begin
  确定问题的表达方式;
  随机生成初始群体;
  评价个体适应度。
```

```
While（终止准则不满足）do
    重组：将两个父代个体交换目标变量和随机因子，产生新个体；
    突变：对重组后的个体添加随机量，产生新个体；
    计算新个体适应度；
    选择：根据选择策略，挑选优良个体组成下一代群体。
End While
输出最优个体。
End
```

GA 与 ES 都是模拟生物界自然进化过程而建立的鲁棒性计算机算法，同属于进化计算。在统一框架下对二者进行比较，可以发现它们有许多相似之处，例如它们的算法中都存在变异算子。GA 与 ES 的不同之处有以下几点。

（1）应用领域不同。GA 从广义上说是一种自适应搜索技术，应用于参数优化、机器学习、免疫系统等许多领域；ES 是一种数值优化的方法，主要应用于离散型优化问题。

（2）求解空间操作不同。GA 是将原问题的解空间映射到串空间之中，然后再施行遗传操作，它强调个体基因结构的变化对其适应度的影响。而 ES 直接在解空间上进行操作，强调进化过程中从父体到后代行为的自适应性和多样性，强调进化过程中搜索步长的自适应性调节。

（3）ES 和 GA 表示个体的方式不同。ES 在浮点矢量上运行，而 GA 一般运行在二进制矢量上。

（4）ES 和 GA 的选择过程不同。标准 GA 对每一个个体都指定一个非零的选择概率；ES 确定地把某些个体排除在被选择之外。

（5）ES 和 GA 的复制参数不同。GA 的复制参数（交叉和变异的可能性）在进化过程中保持恒定，而 ES 时时改变它们。ES 中变异作为主要搜索算子，而标准 GA 中，变异只处于次要位置。

1.2.5　差分进化算法

差分进化（Differential Evolution，DE）算法是一种新兴的进化计算技术，或称为差分演化算法、微分进化算法、微分演化算法、差异演化算法。它是由 Storn 等人于 1995 年提出的，最初的设想是用于解决切比雪夫多项式问题，后来发现 DE 算法也是解决复杂优化问题的有效技术[13]。DE 算法与人工生命，特别是进化算法有着极为特殊的联系。DE 算法是基于群体智能理论的优化算法，通过群体内个体间的合

作与竞争产生的群体智能指导优化搜索。但相比于进化算法，DE 算法保留了基于种群的全局搜索策略，采用实数编码基于差分的简单变异操作和一对一的竞争生存策略，降低了遗传操作的复杂性。同时，DE 算法特有的记忆能力使其可以动态跟踪当前的搜索情况，以调整其搜索策略，具有较强的全局收敛能力和鲁棒性，且不需要借助问题的特征信息，适于求解一些利用常规的数学规划方法所无法求解的复杂环境中的优化问题。

DE 算法是一种基于群体进化的算法，具有记忆个体最优解和种群内信息共享的特点，即通过种群内个体间的合作与竞争来实现对优化问题的求解，其本质是一种基于实数编码的具有保优思想的贪婪遗传算法。DE 算法首先在问题的可行解空间随机初始化种群，对当前种群进行变异和交叉操作产生另一个新种群，然后利用基于贪婪思想的选择操作对这两个种群进行一对一的选择，从而产生最终的新一代种群。DE 算法通过利用种群中个体的距离和方向信息来寻找全局最优解。为此，种群中的每个个体通过动态改变差分项的方向和步长来调整空间的搜索行为。在每一代中，变异和交叉操作均作用于个体，以此产生一个新的种群。然后利用选择操作在这新旧种群中进行比较，来选择产生下一代种群。标准 DE 算法流程流程如下所示。

Begin

随机初始化 DE 的种群;

根据适应度函数，评价个体;

确定具有最好目标值的 X_{best}。

While（终止准则不满足）do

对每个个体执行变异操作（Mutation），以获得相应的变异个体:

$$V_i(t+1) = X_{\text{best}}(t) + F[X_{r1}(t) - X_{r2}(t)]$$

对个体和其变异个体执行交叉操作（Crossover），以获得试验个体;

$$u_{i,j}(t+1) = \begin{cases} v_{i,j}(t+1) & \text{若 rand}(j) \leqslant C_R \text{ 或 } j=\text{rand}(i)，C_R \text{ 为交叉因子} \\ x_{i,j}(t) & \text{其他; } j=1,2,\cdots,d \end{cases}$$

根据适应度函数，评价试验个体的目标值;

在个体和其试验个体之间进行选择操作（Selection），以生成新个体;

$$X_i(t+1) = \begin{cases} U_i(t+1) & \text{若} F[U_i(t+1)] < F[X_i(t)] \\ X_i(t) & \text{其他} \end{cases}$$

更新 X_{best}。

End While

输出 X_{best} 及其目标值。

End

DE 算法的搜索性能取决于算法全局探索和局部开发能力的平衡，而这在很大程度上依赖于算法的控制参数的选取，包括种群规模缩放比例因子和交叉因子等。相对其他进化算法而言，DE 算法所需调节的控制参数较少，合理的参数设置可以参考相关文献。归纳起来，DE 算法具有如下优点。

（1）算法通用，不依赖于问题信息。

（2）算法原理简单，容易实现。

（3）群体搜索，具有记忆个体最优解的能力。

（4）协同搜索，具有利用个体局部信息和群体全局信息指导算法进一步搜索的能力。

（5）易于与其他算法混合，构造出具有更优性能的算法。

DE 算法性能优越、容易理解、易于实现，一经提出就备受关注并得到了广泛的应用，但是 DE 算法本身仍有很多值得研究的地方，例如参数的设置问题。DE 算法的性能很大程度上和参数的选取有关，比较重要的有群体规模、缩放因子和交叉因子。

1．群体规模的选择

从计算复杂度分析，种群规模越大，搜索到全局最优解的可能性就越大，然而所需的计算量和计算时间也要增加。而且最优解的质量并非随种群规模的增大一味变好，有时种群规模的增大，反而会使最优解的精度降低，因此，合理选取种群规模对算法的搜索效率的提高具有重要意义。

当种群规模增大到一定个数时，解的精度不再提高，甚至会出现降低的情况。这是因为较大的种群规模能保持种群的多样性，但会降低收敛速度，多样性和收敛速度必须保持一定的平衡。因此，当种群规模太大时，如果不增加最大进化代数，精度反而会降低。另外，种群规模越大，多样性就越大，所以如果种群过早收敛，就要增加种群规模以增加多样性。

在给定最大进化代数情况下，对低维简单问题，种群规模在[15，35]之间较好；对高维复杂问题，种群规模在[30，50]之间时，优化结果较好。总之，在给

定最大进化代数下，种群规模在[15，50]之间时，能很好地保持多样性和收敛速度的平衡。

2．缩放因子*F*对优化效果的影响

缩放因子 F 取值[0.5，1]之间时，算法得到的结果较好；当 $F<0.5$ 或 $F>1$ 时，算法求得的解的质量不高。缩放因子 F 是用于控制差分向量对变异个体 V 的影响的。当 F 较大时，差分向量对 V 的影响较大，能产生较大的扰动，从而有利于保持种群的多样性；反之，F 较小时，扰动较小，缩放因子能起到局部精细化搜索的作用。因此，F 对种群的多样性起到了一定的调节作用。缩放因子 F 取值太大，虽然能保持种群多样性，但算法近似随机搜索，搜索效率低下，求得的全局最优解精度低；反之，F 太小，种群多样性丧失很快，算法易于陷入局部最优出现早熟收敛。

由于 DE 算法是一种“贪婪”选择算法，所以，随着种群的不断进化，各个个体逐渐靠近最优个体，个体间的差异也会逐渐减小。这就意味着，当算法进化到一定程度时，种群多样性就会丧失。种群多样性对算法的全局搜索能力有一定影响。种群多样性大，增加了从局部最优逃脱的可能性，有利于全局搜索，但会降低收敛速度；种群多样性小，有利于局部搜索，收敛速度快，但是，易于陷入局部最优，出现所谓的早熟现象。

综上，F 对算法的局部搜索和全局搜索起到了一定的调节作用。F 较大，有利于保持种群多样性和全局搜索；F 较小，有利于局部搜索和提高收敛速度。

3．交叉因子C_R对优化效果的影响

C_R 的值较小时，所需的函数评价次数较大，收敛速度较慢，但成功率较高，算法的稳定性好；C_R 的值较大时，常常会加速收敛，但易于陷入局部最优，发生早熟现象，达到给定精度的成功率低，稳定性差。可见，成功率和收敛速度是一对矛盾。因此，为了同时保证较高的成功率和较快的收敛速度，对于单峰函数，C_R 取值相对较大些，在[0.6，0.8]区间；对其他复杂、多峰函数 C_R 取值应相对小些，在[0.1，0.5]区间。

新子代个体 U 是由变异个体 V 和父代个体 X 分量间相互交叉而产生的。C_R 的值越大，则 V 对 U 的贡献越多，有利于开拓新空间，从而加速收敛，但在后期变异个体趋于同一，不利于保持多样性，从而易于“早熟”，稳定性差；C_R 的值越小，则 X 对 U 的贡献越多，这样就减弱了算法开拓新空间的能力，收敛速度相对较慢，但有利于保持种群多样性，从而能有较高的成功率。

1.3　群体智能

群体智能（Swarm Intelligence，SI）就是无智能或具有简单智能的个体在无集中控制的情况下，通过单个个体自身的简单行为，使得整个群体表现出某种智能行为，从而解决某些特定的问题。

1.3.1　群体智能概述

SI 的概念最早由 Beni、Hackwood 和 Wang 在分子自动机系统中提出的[14,15]。分子自动机中的主体在一维或二维网格空间中与相邻个体相互作用，从而实现自组织。1999 年，Bonabeau、Dorigo 和 Theraulaz 在他们的著作《Swarm Intelligence: From Natural to Artificial Systems》中对 SI 进行了详细的论述和分析，给出了 SI 的一种不严格定义：任何一种由昆虫群体或其他动物社会行为机制而激发设计出的算法或分布式解决问题的策略均属于 SI[16]。

这里，群体（Swarm）可被描述为一些相互作用相邻个体的集合体，鱼群、蜂群、蚁群、鸟群都是群体的典型例子。鱼聚集成群可以有效地逃避捕食者，因为任何一只鱼发现异常都可带动整个鱼群逃避。蚂蚁成群则有利于寻找食物，因为任一只蚂蚁发现食物都可带领蚁群来共同搬运和进食。一只蜜蜂或蚂蚁的行为能力非常有限，它几乎不可能独立存在于自然世界中，而多个蜜蜂或蚂蚁形成的群体则具有非常强的生存能力，且这种能力不是多个个体之间能力通过简单叠加所获得的。社会性动物群体所拥有的这种特性能帮助个体很好地适应环境，个体所能获得的信息远比它通过自身感觉器官所取得的多，其根本原因在于个体之间存在着信息交互能力。信息的交互过程不仅仅在群体内传播了信息，而且群内个体还能处理信息，并根据所获得的信息（包括环境信息和附近其他个体的信息）改变自身的一些行为模式和规范，这样就使得群体涌现出一些单个个体所不具备的能力和特性，尤其是对环境的适应能力。这种对环境变化所具有适应的能力可以被认为是一种智能（关于适应性与智能之间的关系存在着一些争议，Fogel 认为智能就是具备适应的能力），也就是说动物个体通过聚集成群而涌现出了智能。因此，Bonabeau 将 SI 的定义进一步推广为：无智能或简单智能的主体通过任何形式的聚集协同而表现出智能行为的特性。这里我们关心的不是个体之间的竞争，而是它们之间的协同。

James Kennedy 和 Russell C.Eberhart 在 2001 年出版了《Swarm Intelligence》，是群智能发展的一个重要里程碑，因为此时已有一些群智能理论和方法得到了应用。他们不反对 Bonabeau 关于 SI 的定义，赞同其定义的基本精神，但反对定义中使用

“主体”一词[17]。其理由是“主体”所带有自治性和特殊性是许多群体的个体所不具备和拥有的，这将大大限制群体的定义范围。他们认为暂时无法给出合适的定义，赞同由 Mark Millonas 提出的构建一个 SI 系统所应满足的五条基本原则[18]。

（1）Proximity Principle：群内个体具有能执行简单的时间或空间上的评估和计算的能力。

（2）Quality Principle：群内个体能对环境（包括群内其他个体）的关键性因素的变化做出响应。

（3）Principle of Diverse Response：群内不同个体对环境中的某一变化所表现出的响应行为具有多样性。

（4）Stability Principle：不是每次环境的变化都会导致整个群体的行为模式的改变。

（5）Adaptability Principle：环境所发生的变化中，若出现群体值得付出代价的改变机遇，群体必须能够改变其行为模式。

《Swarm Intelligence》最重要的观点是“Mind is Social”[17]，也就是认为人的智能是源于社会性的相互作用，文化和认知是人类社会性不可分割的重要部分，这一观点成了群体智能发展的基石。SI 已成为有别于传统人工智能中连接主义、行为主义和符号主义的一种新的关于智能的描述方法。SI 的思路为在没有集中控制且不提供全局模型的前提下寻找复杂的分布式问题求解方案提供了基础。在计算智能领域已取得成功的两种基于 SI 的优化算法是蚁群优化算法和粒子群优化算法。目前，已有的基于 SI 的优化算法都是源于对动物社会通过协作解决问题行为的模拟，它主要强调对社会系统中个体之间相互协同作用的模拟。这一点与 EC 不同，EC 是对生物演化中适者生存的模拟。

与 EC 一样的是，SI 的目的并不是为了忠实地模拟自然现象，而是利用他们的某些特点去解决实际问题。另一个与 EC 的相同点是，基于 SI 的优化算法也是概率搜索算法。目前，已有的 SI 理论和应用研究证明 SI 方法是一种能够有效解决大多数优化问题的新方法，更重要的是，SI 潜在的并行性和分布式特点为处理大量的以数据库形式存在的数据提供了技术保证。

无论是从理论研究还是应用研究的角度分析，SI 理论及应用研究都是具有重要学术意义和现实价值的。由于 SI 理论依据是源于对生物群落社会性的模拟，因此其相关数学分析还比较薄弱，这就导致了现有研究还存在一些问题[19]。首先，SI 算法的数学理论基础相对薄弱，缺乏具备普遍意义的理论性分析，算法中涉及的各种参数设置一直没有确切的理论依据，通常都是按照经验方法确定，对具体问题和应用环境的依赖性比较大。其次，同其他的自适应问题处理方法一样，SI 也不具备绝对

的可信性，当处理突发事件时,系统的反应可能是不可测的，这在一定程度上增加了其应用风险。

1.3.2　蚁群优化算法

蚁群优化（Ant Colony Optimization，ACO）算法由 Colorni、Dorigo 和 Maniezzo 于 1991 年提出的[20,21]，它是通过模拟自然界蚂蚁社会寻找食物的方式而得出的一种仿生优化算法。自然界中蚁群寻找食物时会派出一些蚂蚁分头在四周游荡，如果一只蚂蚁找到食物，它就返回巢中通知同伴并沿途留下“信息素”（Pheromone）作为蚁群前往食物所在地的标记。信息素会逐渐挥发，如果两只蚂蚁同时找到同一食物，又采取不同路线回到巢中，那么比较绕弯的一条路上信息素的气味会比较淡，蚁群将倾向于沿另一条更近的路线前往食物所在地。ACO 算法设计虚拟的“蚂蚁”，让它们摸索不同路线，并留下会随时间逐渐消失的虚拟“信息素”。根据“信息素较浓的路线更近”的原则，即可选择出最佳路线。

ACO 算法可以看成是一种基于解空间参数化概率分布模型（Parameterized Probabilistic Model）的搜索算法框架，其中解空间参数化概率模型的参数就是信息素，因而这种模型就是信息素模型。在基于模型的搜索算法框架中，可行解通过在一个解空间参数化概率分布模型上的搜索产生，此模型的参数用以前产生的解来进行更新，使得在新模型上的搜索能集中在高质量的解搜索空间内。这种方法的有效性建立在高质量的解总是包含好的解构成元素的假设前提下。通过学习这些解构成元素对解的质量的影响有助于找到一种机制，并通过解构成元素的最佳组合来构造出高质量的解。一般来说，一个基于模型的搜索算法通常使用以下两步迭代来解决优化问题。

（1）可行解通过在解空间参数化概率分布模型上的搜索产生。

（2）用搜索产生的解来更新参数化概率模型，即更新解空间参数化概率分布的参数，使得在新模型上的搜索能集中在高质量的解搜索空间内。

在 ACO 算法中，基于信息素的解空间参数化概率模型（信息素模型）以解构造图的形式给出。在解构造图上，定义了一种作为随机搜索机制的人工蚂蚁，蚂蚁通过一种分布在解构造图上被称为信息素的局部信息的指引，在解构造图上移动，从而逐步地构造出问题的可行解。信息素与解构造图上的节点或弧相关联，作为解空间参数化概率分布模型的参数。由于旅行商问题（Traveling Salesman Problem，TSP）能够非常直接地映射为解构造图（城市为节点，城市间的路径为弧，信息素分布在弧上），且是一个 NP 难题，因而 ACO 算法的大部分成果都集中在 TSP 问题上。一般而言，用于求解 TSP 问题、生产调度问题等组合优化问题的 ACO 算法都遵循如下的统一算法流程。

```
Begin
  设置参数，初始化信息素踪迹。
  While（不满足终止准则）do
    For 蚁群中的每只蚂蚁
      For 每个解构造步（直到构造出完整的可行解）
        蚂蚁按照信息素及启发式信息的指引构造一步问题的解;
        进行信息素局部更新。（可选）
      End For
    End For
    （1）以某些已获得的解为起点进行邻域（局部）搜索;（可选）
    （2）根据某些已获得的解的质量进行全局信息素更新。
  End While
End
```

在算法中，蚂蚁逐步构造问题的可行解，在一步解的构造过程中，蚂蚁以概率方式选择信息素强且启发因子高的弧到达下一个节点，直到不能继续移动为止。此时蚂蚁所走过的路径对应求解问题的一个可行解。局部信息素更新针对蚂蚁当前走过的一步路径上的信息素进行，全局信息素更新是在所有蚂蚁找到可行解之后，根据发现解的质量或当前算法找到的最好解对路径上的信息素进行更新。总的来说，ACO 算法的主要特点可以概括为以下几点。

（l）采用分布式控制，不存在中心控制。

（2）每个个体只能感知局部的信息，不能直接使用全局信息。

（3）个体可以改变环境，并通过环境来进行间接通信。

（4）具有自组织性，即群体的复杂行为是通过个体的交互过程中突现出来的智能。

（5）是一类概率型的全局搜索方法，这种非确定性是算法能够有更多的机会求得全局最优解。

（6）其优化过程不依赖于优化问题本身的严格数学性质，比如连续性、可导性以及目标函数和约束函数的精确数学描述。

（7）是一种基于多主体（Multi-Agent）的智能算法，各主体之间通过相互协作来更好的适应环境。

（8）具有潜在的并行性，其搜索过程不是从一点出发，而是同时从多个点同时进行。这种分布式多智能体的协作过程是异步并发进行的，分布并行模式将大大提高整个算法的运行效率和快速反应能力。

1.3.3　粒子群优化算法

粒子群优化（Particle Swarm Optimization，PSO）算法是模拟鸟类捕食行为的群体智能算法，由美国电气工程师 Eberhart 和社会心理学家 Kennedy 于 1995 年提出[22]。由于 PSO 算法容易实现，需要调整的参数少，一经提出就受到了研究者的重视，被广泛应用到各个领域。有关 PSO 算法的概念、原理和应用将在后续章节详细介绍。

1.3.4　其他智能算法

人工鱼群算法也是一个模拟自然界生物特性的智能算法，它是由浙江大学的李晓磊[23]等人在 2002 年首次提出以来，并得到了国内外学者的广泛关注。目前对该算法的研究应用已经渗透到多个应用领域，由最初的求解一维静态优化问题已发展到解决多维动态组合优化问题。在李晓磊的博士论文中，给出了人工鱼群算法的原理和算法步骤的详细描述，并对算法的收敛性能和算法中控制参数对算法性能的影响进行了分析。针对组合优化问题，提出了人工鱼群算法中的距离、邻域和中心等概念，并给出了算法在组合优化问题应用中的描述；针对大规模系统的优化问题，给出了基于分解协调思想的人工鱼群算法，并介绍了人工鱼群算法中常用的一些改进方法，以及人工鱼群算法在时变系统的在线辨识和 PID 参数整定中的应用分析；最后指出了鱼群模式和算法的今后研究发展方向。目前，人工鱼群算法已经成为交叉学科中一个非常活跃的前沿性研究问题。

人口迁移算法（Population Migration Algorithm，PMA）是我国学者周永华和毛宗源于 2003 年提出的一类模拟人口迁移机理的全局优化算法[24]，不仅具有通用、简单、并行、稳定等优点，而且还具有较强的全局搜索能力。与以往的模拟进化算法不同，人口迁移算法是建立在对人口移动规律的整体认识的基础上，对人口迁移的简单模拟。算法思想来源于社会领域中人口随经济重心而转移、随人口压力增加而扩散的规律，即模拟的是人往高处走，人往富处流，当某个优惠地区的相对人口压力增加时，人们就会迁出该优惠地区去寻找更好更适合自己的优惠地区的这种规律，这为人们进行算法设计提供了一种新的思路与方向。已有数值实验和应用说明人口迁移算法具有良好的全局搜索能力，但理论基础以及广泛的实际应用仍有待深入研究。

人工蜂群算法（Artificial Bee Colony Algorithm，ABCA）是 Karaboga 于 2005 年提出的一种新颖的 SI 优化算法[25]，主要模拟蜂群的智能采蜜行为，蜜蜂根据各自的分工进行不同的采蜜活动并实现蜜源信息的共享和交流从而找到问题的最优解。

在 ABCA 中，人工蜂群包含 3 个组成部分：采蜜蜂、观察蜂和侦察蜂。群体的一半由采蜜蜂构成，另一半由观察蜂构成；每一处蜜源仅仅有一个采蜜蜂，也就是说蜜源数和采蜜蜂的数目相等；放弃所采蜜源的采蜜蜂成为侦察蜂。搜索活动可以概括如下：采蜜蜂根据它们记忆中的蜜源位置在其邻域内确定另一个蜜源，并在蜂巢内将它们的信息通过舞蹈共享给观察蜂，观察蜂选择其中的一个蜜源；观察蜂根据所选择的蜜源在其邻域内搜索另一个蜜源；放弃所采蜜源的采蜜蜂将成为侦察蜂，并搜索一个新的随机蜜源。

在 ABCA 算法中每个蜜源的位置代表优化问题的一个可能解，蜜源的花蜜量对应于相应解的质量或适应度。ABCA 算法首先随机产生初始群体，即 n 个初始解，n 为采蜜蜂数也等于蜜源数目。每个解 $\boldsymbol{X}_i$（$i=1,2,\mathrm{L}\ ,n$）是一个 d 维的向量。以这些初始解为基础采蜜蜂、观察蜂和侦察蜂开始进行循环搜索。采蜜蜂根据它记忆中的局部信息产生一个变化的位置，并检查新位置的花蜜量。如果新位置优于原位置，则该蜜蜂记住新位置并忘记原位置。所有的采蜜蜂完成搜索过程后，它们将所知道的蜜源信息通过舞蹈与观察蜂共享，观察蜂根据从采蜜蜂处得到的信息，按照与花蜜量相关的概率选择一个蜜源位置，并像采蜜蜂那样对记忆中的位置做一定的改变，并检查新候选位置的花蜜量。若新位置优于记忆中的位置，则用新位置替换原来的蜜源位置，否则保留原位置。换句话说“贪婪选择”机制被用于选择原位置和新的候选位置。

参 考 文 献

[1] Reeves C.R. Modern Heuristic Techniques for Combinatorial Problems. Oxford: Blackwell Scientific Publications, 1993.

[2] Wolpert D.H, Macready W.G. No Free Lunch Theorems for Search. Technical Report SFI-TR -95-02-010, Santa Fe Institure.1995:1～32.

[3] Wolpert D.H, Macready W.G.No Free Lunch Theorems for Optimization. IEEE TRANSACTION ON EVOLUTIONARY COMPUTATION,1997,1(1):67～82.

[4] Christensen S, Oppacher F. What Can We Learn from No Free Lunch?A First Attempt to Characterize the Concept of Searchable Function.The 2001 Genetic and Computation Conference. Piscataway, NJ:IEEE Press, 2001(12):19～1226.

[5] Fogel D.B. Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. NewYork: Wiley-IEEE Press,1995.

[6] Holland J.H. Adaptation in Natural and Artificial Systems. Ann Arbor:University

of Michigan.

[7] Goldberg D.E. Genetic algorithms in Search, Optimization, and Machine Learning. Reading MA: Addison-Wesley,1989.

[8] Goldberg D.E, Segrest P. Finite Markov Chain Analysis of Genetic Algorithms.In: The Internationa Conference on Genetic algorithms.Piscataway, NJ:IEEE Service Center,1987.1～8.

[9] Cerf R.Asymptotic Convergence of Genetic Algorithms.ADVANCES IN APPLIED PROBABILITY, 1998, 30(2):521～550.

[10] Leung Yee, Gao Yong,Xu Zong-Ben. A Markov Chain Analysis of Premature Convergence in Genetic Algorithms. In:The International Conference on Neural Information Processing, Springer Verlag, Vol(2),1996.1330～1334.

[11] Fogel L, Owens A and Walsh M. Artificial Intelligence through Simulated Evolution.New York:John Wiley,1966.

[12] Rechenberg I.Bionic. Evolution and Optimizing.Naturwissen Schaftliche Rundschau, 26(11): 465～472.

[13] Storn R and Price K.Differential Evolution-A Simple and Efficient Heuristic for Global Optimization Over Continuous Spaces, Journal of Global Optimization,11(4), 341～359.

[14] Hackwood S,Wang J.The Engineering of Cellular Robotic Systems.In:IEEE International Symposium on Intelligent Control.Piscataway,NJ:IEEE Press,1988.70～75.

[15] Hackwood S,Beni G.Self-organization of Sensors for Swarm Intelligence.In:IEEE International conference on Robotics and Automation.Piscataway,NJ:IEEE Press, 1992.819～829.

[16] E.Bonabeau, M.Dorigo, and G.Theraulaz.Swarm Intelligence:From Natural to Artificial Systems. New York: Oxford University Press, 1999.

[17] Kennedy James F, Eberhart Russell and Shi Yuhui. Swarm Intelligence, Elsevier Science Ltd, 2005.

[18] Millonas M.M. Swarms,Phase Transitions and Collective Intelligence. In: Langton, C.G.(Ed.), Artificial Life III.Santa Fe Institute Studies in the Sciences of Complexity,Vol.XVII. Addison- Wesley,1994:417～445.

[19] Mark Fleischer.Foundations of Swarm Intelligence:From Principles to Practice. In:Conference on Swarming:Network Enabled C4ISR.E-Print Alert Service.2003.1～13.

[20] Colorni A, Dorigo M and Maniezzo V.Distributed Optimization by Ant Colonies. The First European conference on Artificial Life.France:Elsevier,1991.134～142.

[21] Dorigo M, Maniezzo V and Colorni A.The Ant System: Optimization by a Colony of Cooperating Agents.IEEE TRANSACTIONS ON SYSTEM,MAN,AND CYBER NETICS—Part B.1996,26:29～41.

[22] Kennedy J,Eberhart R.C.Particle Swarm Optimization.In:IEEE International Conference on Neural Networks,IV.Piscataway,NJ:IEEE Service Center，1995.1942～1948.

[23] 李晓磊，邵之江，钱积新．一种基于动物自治体的寻优模式：鱼群算法．系统工程理论与实践，2002，22(11)：32～38.

[24] 周永华，毛宗源一种新的全局优化搜索算法一人口迁移算法．华南理工大学学报（自然科学版），31（3），2003（3）：l～5.

[25] D.Karaboga, An Idea Based On Honey Bee Swarm For Numerical Optimization, Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

第 1 章　概　　论

优化理论与方法是一门应用性很强的学科，用于研究某些基于数学描述问题的最优解。美国工程院院士哈佛大学何毓琦（Yu-Chi Ho）教授指出“任何控制与决策问题本质均可以归结为优化问题”。工程中很多的实际问题在进行数学建模后，都可以抽象为一个组合优化问题。通过求解该类问题，可以为决策者提供最佳选择或最佳信息，即针对给出的实际问题，从众多的方案中选出最佳方案。优化问题最早可以追溯到古希腊时代的极值问题，如谷物的堆砌问题、等周问题等。但由于缺乏合适的计算工具和系统的理论指导，一直没有得到应有的发展。优化成为一门独立的学科是在 20 世纪 40 年代末，一方面，需要为实际生产中涌现的复杂优化问题提供快速而实用的优化算法；另一方面，包括泛函分析在内的数学理论的发展也进一步奠定了优化方法的理论基础。而计算机的出现为各种优化算法的快速实现提供了更为便捷的计算工具，这些因素促使优化逐渐成为一门应用广泛、生机勃勃的学科。

1.1　优化技术

优化是人们在工程技术、科学研究和经济管理等诸多领域中经常遇到的问题，在计算机科学、人工智能、运筹学等领域中占有非常重要的地位，是指在合理的时间范围内为一个优化问题寻找最优可行解的过程，其中优化问题的可行解之间是可以进行量化比较的。

1.1.1　优化技术介绍

具体来说，最优化问题就是在给定的约束条件下，寻找一组参数值，以使系统（或函数）的某些最优性度量得到满足，使系统（函数）的某些性能指标达到最大或最小。寻求问题最优可行解过程的第一步是要对问题进行描述和建立问题的数学模型，即用数学方程式和不等式来描述说明所求的最优化问题，其中包括目标函数和约束条件，而识别目标、确定目标函数的数学表达形式尤为关键。

1．变量的确定

变量是最优化问题或系统中待确定的某些关键元素。变量数和约束条件的多少

直接决定优化问题的规模大小，一般工程上最优设计问题属于中小规模的优化问题，而生产计划、调度问题中变量数可达几百个、几千个，属于大规模优化问题。

2．约束条件

目标函数求解时的某些限制称为约束，如变量取值范围的限定、可用资源的有限性以及所求问题的技术标准要求，此外还应满足物理系统的基本方程和性能方程。给出的约束条件越接近实际系统，则所求得的最优化问题的解也越接近于实际的最优解。约束条件可分为等式约束和不等式约束。

3．目标函数

最优化是有一定的标准或评价方法的，而目标函数就是评价优化效果的标准的数学描述，用 $f(x)$表示。最优化常指最小化和最大化两类问题。由于函数的最大化等价于其负的最小化，所以最小化和最大化问题在实际求解过程中并没有本质上的区别，本书中如不作特殊说明，一般指最小化问题。最优化问题的一般形式为

$$\begin{cases} \min f(x) \\ \text{s.t.} \quad g(x)=0 \\ \quad\quad h(x)\geqslant 0 \\ \quad\quad x\in \mathrm{A} \end{cases} \tag{1.1}$$

其中 x 为决策变量，A 为解的可行域，$f(x)$为目标函数，$g(x)=0$ 为等式约束，$h(x)\geqslant 0$ 为不等式约束。使目标函数 $f(x)$取得最优值的解称为最优解，记为

$$x^*\in a,\ \text{s.t.}\quad f(x^*)\leqslant f(x)\qquad \forall\, x\in a \tag{1.2}$$

对优化问题的分类有许多种，据不同的观点可以分为不同的类型，通常有如下不同分类方法。

（1）按是否有约束分类：取决于问题中有无约束，可分为有约束问题和无约束问题两种。

（2）按目标函数及约束函数特性分类；可分为线性规划、非线性规划、几何规划、整数规划和二次规划问题等。从计算的观点来说，这种分类法很有用，因为通常研究出的很多种方法都是对某一类问题有效。

（3）按计算复杂性分类：可分为 P 问题、NP 问题、NP-Hard 问题、NPC 问题等。

（4）按所包含变量确定性的性质分类：可分为确定性规划问题和随机规划问题。

（5）按目标函数与约束函数的可分离性分类：可分为不可分离问题和可分离问题。

下面介绍几个在优化问题中经常用到的概念，以帮助更好地理解后面的内容。

1．解之间的距离测度函数

设〈*A*,*f*〉是某优化问题的一个实例，定义 $\mathrm{Dist}: A\times A \to R^+$ 为计算该优化问题中的两个解之间的距离测度函数。距离测度函数的定义与优化问题决策变量的表示有很大关系，与优化算法的性能也有非常大的关系。

2．解的邻域

设〈*A*,*f*〉是某优化问题的一个实例，Dist 为解之间距离测度函数。*A* 上的一个映射 $N_e: x\in A \to N_e(x)\in 2^A$ 成为邻域映射，其中 2^A 表示 *A* 的所有子集组成的集合。也就是对任意一个 $v\in A$，集合 $N_e(v)\subseteq A$ 被称为 *v* 的邻域，$N_e: x\in N_e(v)$ 称为 *v* 的一个邻居。对于任意给定 $\varepsilon\in R^+$，N_e 的数学描述为：

$$\begin{aligned} N_e &: A \to 2^A \\ v &\to \{x\in A \mid \mathrm{Dist}(x,v)\leqslant \varepsilon\} \end{aligned} \tag{1.3}$$

邻域的构造也依赖于问题决策变量的表示，领域的结构在优化算法中起着非常重要的作用。有了邻域的定义以后，就可以定义局部、全局最优的概念了。

3．局部最优

设〈*A*,*f*〉是某优化问题的一个实例，N_e 为邻域函数。对于确定的 N_e，若 N_e 满足 $f(x^*)\leqslant f(x), \forall x\in N_e(x^*)$，则称 N_e 为在 *A* 上局部最优。

4．全局最优

设〈*A*,*f*〉是某优化问题的一个实例。若 N_e 满足 $f(x^*)\leqslant f(x), \forall x\in A$，则称 N_e 为在 *A* 上全局最优。

5．可接受解

设〈*A*,*f*〉是某优化问题的一个实例，N_e 为在 *A* 上局部最优。对于给定 $\varepsilon\in R^+$，集合 $C=\{x\in A: \mid f(x)-f(x^*)\mid \leqslant \varepsilon\}$ 被称为可接受解的集合。可接受解在优化问题中是非常重要的，因为在非常多的实例中，有限的时间内保证搜索到全局最优几乎是不可能的。在这种情况下，优化的目的往往是搜索一个满足条件的可以接受解。

1.1.2 优化算法

在某种意义上，所谓优化算法其实就是一种搜索过程和规则，它是基于某种思想和机制，通过一定的途径和规则求得满足用户要求的解的过程。优化的算法非常多，大致分类如图 1.1 所示，基本可分为两大类：精确算法和启发式算法。其中精确算法可对解空间进行彻底搜寻，得到全局最优解，如线性规划、整数规划、动态规划等算法。但这类算法所适用的优化问题很有限，对 NP-Hard、NPC 等类问题则通常使用启发式算法，又称近似算法，例如，邻域搜索算法和基于系统动态演示的算法等。

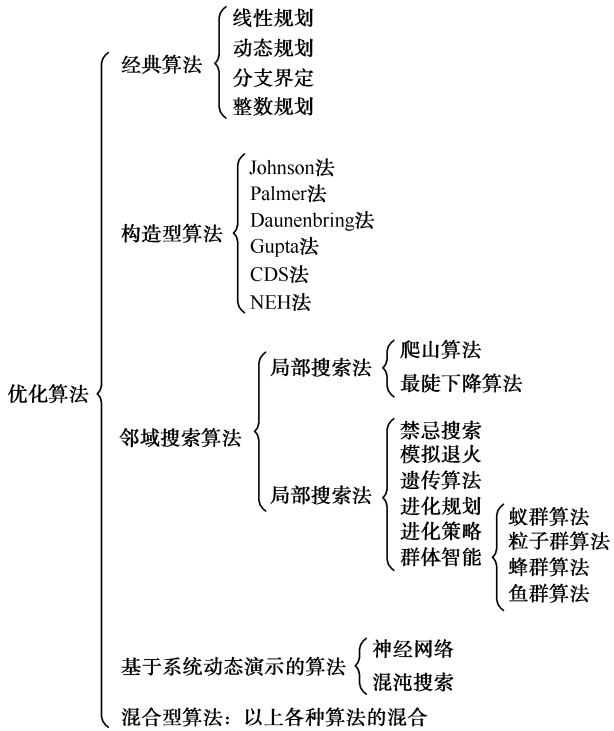

图 1.1 优化算法的组成及分类

Reeves 对启发式算法作如下定义[1]：启发式算法是一种技术，这种技术使得在可接受的计算费用内寻找好的解，但不一定能保证所得解的可行性和最优性，甚至在多数情况下，无法阐述所得解同最优解的近似程度。

根据 Reeves 的定义可知，启发式算法在多数情况不能给出最坏解的偏差程度。20 世纪 60 至 70 年代，由于人们对数学模型和最优解算法的重视，一些根据实际问题而提

出的启发式算法被称为“快速而丑陋”的算法，因此一直得不到重视。随着算法复杂性理论的完善，人们不再强调一定要得到最优解，启发式算法才开始得到广泛的重视和发展。目前优化算法的研究主要集中在启发式算法上，包括禁忌搜索、模拟退火算法、演化计算等，本书所讨论的粒子群优化（PSO）算法也属于启发式算法。当然，也有一些研究者将精确算法和启发式算法进行混合。通常，启发式优化算法给出的解是可接受解，是全局最优解的上界，因此也可被称为上界算法。也有一些算法可以获得下界，例如线性规划松弛算法、拉格朗日松弛算法等，通常被称为下界算法。

在优化理论研究领域中，一个最有趣的研究成果是 Wolpert 和 Macready 于 1997 年在《IEEE Transactions on Evolution Computation》上发表的题为“No Free Lunch Theorems for Optimization”的论文[2,3]。在这篇论文中，提出并严格论证了所谓的“无免费午餐定理”，简称 NFL 定理，NFL 定理可以描述如下：对任意给定的两种算法 A 和 B，则对于所有可能的问题集，它们的平均性能是相同的（衡量性能的指标有多种，如最优解、收敛率等）。也就是说对于所有可能的优化问题，如果 A 在某些问题类上表现比 B 好（差），那么就一定存在其他一些问题类，A 在其上的表现比 B 差（好），即 A、B 对所有问题的平均表现度量是完全一样的。

该定理表明对所有可能函数的集合而言，任何优化搜索方法的性能都是等价的，也就是说没有一个优化搜索方法是优于其他优化搜索方法的，甚至没有其他任何算法能够比搜索空间的线性列举或者纯随机搜索算法更优。那么人们就会产生这样的疑问，既然 NFL 定理表明进化算法并不比一般随机搜索算法性能好，可为什么还要对进化算法进行研究和改进呢？实际上，定理本身只是否定了去寻找一个通用的最优算法的可能性，而对于一些特定的函数集合，NFL 定理则认为存在一个适用于该集合上的优化算法，其性能在该集合上优于其他算法。在求解某些或某个特定的复杂优化问题时，可能会出现现有大多数优化方法都不适用、而适用的少数方法效果又不理想的情况，可以考虑使用进化算法。由于进化类算法本身具有很强的适用性，且对目标函数的连续性无任何要求，因此把进化算法作为求解复杂优化问题的候选算法是非常有现实意义的。

NFL 定理的主要价值在于它对研究与应用优化算法时的观念性启示作用。虽然 NFL 定理是在许多假设条件下得出的，但它仍然在很大程度上反映出了优化算法的本质。NFL 能成立的关键在于“所有可能函数的集合”，这一断言在欺骗性的和随机的函数上是成立的。当所面对的是一个大的而且形式多样的适应度值函数类时，就必须考虑算法间所表现出的 NFL 效应。由此，Christensen 等人提出了“可搜索的函数”的定义，以及适用此类函数集的一个通用的算法，可证明该算法在可搜索的函数集上的性能优于随机搜索[4]。

因此，对于所有函数集合，并不存在万能的最佳算法，所有算法在整个函数类上的平均表现度量是一样的。基于上述观点，关于优化算法的研究就应该从寻找所有可能函数类上的通用优化算法转变为以下两方面。

（1）以算法为导向，确定其适用的问题类。对于每一个算法，都有其适用和不适用的问题；对于给定的算法，尽可能通过理论分析和实际应用，找出其适用的范围，归纳特定的问题类，使其成为一个指示性的算法。

（2）以问题为导向，确定其适用的算法。对于小的特定问题类，或者一个特定的实际应用问题，设计出针对性的适用算法。实际上，大多数在优化算法方面的研究工作都是属于这一范畴的，因为它们主要是根据进化的原理设计新的算法，或者将现有算法进一步优化，以期对若干特定的函数类取得较好的优化效果。

1.2 进化计算

自 20 世纪 50 年代中期创立仿生学以来，人们从生物进化的机理中受到启发，提出了许多用于解决复杂优化问题的新方法，如遗传算法（Genetic Algorithm，GA）、进化策略 （Evolution Strategies，ES）、进化规划（Evolutionary Programming，EP）和差分进化算法（Differential Evolutionary，DE）等，这些方法被广泛应用于科学研究和实际问题求解，并取得了较好的效果。这些方法都是基于生物进化的基本思想来设计、控制和优化人工系统的，因此，这类计算方法一般统称为进化计算（Evolutionary Computation，EC）[5]。它们各具特点，分别代表了进化计算的不同侧面。

1.2.1 进化计算框架

进化算法是受生物进化论和遗传学等理论启发而形成的求解优化问题的一种随机算法，与普通的搜索算法一样属于迭代算法，即从给定的初始解中通过不断地迭代，逐步改进收敛到最优解。在进化计算中，每一次迭代被看成是一代生物个体的繁殖与进化，称为“代”（Generation）。每一个解被称为一个“个体”（Individual），用一组有序排列的基因（Gene）来表示，而每一组解被称为“人口”或“种群”（Population）。在搜索过程中则利用随机性和结构化的信息，使最满足目标的个体获得最大的生存可能，是一种概率型的算法。在自然界中，物种的性质是由染色体决定的，而染色体则是由基因有序地排列组成的。在搜索问题中，目标是由决策变量确定的，决策变量则是由一系列的分量组成的。进化算法正是人为建立并充分利用了这种相似性。

虽然进化计算的不同分支算法实现上有一些差别，但它们具有一个共同特点，即都是借助生物演化的思想和原理来解决问题。作为一种仿生的概率算法，进化算

法可以有效地对解空间进行搜索，而且算法具有较强的通用性，对问题的具体形式和领域知识依赖性不强，同时又有着本质的并行性。因此能够较快地寻找到最优解或满意解。一般而言，广义的进化算法的计算过程如下所示。

```
Begin
    首先初始化一组解。
    While（终止准则不满足）do
      根据满足目标的优劣程度来评价解的性能;
      根据所得解的性能，从当前这组解中选择一定数量的解作为进化后的解的基础;
      对所得到的解进行重组和变异等操作，结果作为进化后的解。
    End While
    输出最优解。
End
```

1.2.2　遗传算法

遗传算法（Genetic Algorithms，GA）是模拟达尔文生物进化论的自然选择和遗传学机理的生物进化过程的计算模型，是一种通过模拟自然进化过程搜索最优解的方法。遗传算法概念最早由 Bagley J.D 在 1967 年提出，是进化计算（EC）中最广为人知的一种。1975 年 Holland 出版了《Adaptation in Natural and Artificial Systems》，被认为是第一本系统论述 GA 的专著[6]。而 1989 年，Holland 的学生 Goldberg 出版的《Genetic Algorithms in Search, Optimization, and Machine Learning》则对 GA 的理论及应用作了全面系统的论述，奠定了现代遗传算法的基础[7]。

GA 的进化对象是由多个个体组成的群体，优化问题的解被表达为个体的染色体，解的值与个体的适应度值相对应，也就是个体的适应能力。群体初始化之后，先基于适应度的概率来选择父代，然后通过交叉重组和变异来维持群体的多样性，如此迭代下去，直到满足终止条件。

GA 中问题解的表示方式也就是个体的染色体，通常是固定长度的串，现在也出现了实数编码的染色体。迭代过程包含三种操作：选择、交叉重组和变异。交叉和变异都按一定的概率进行，通常被称为交叉概率 P_c 和变异概率 P_m，通常 P_c 很高而 P_m 很低。

在选择时，以适应度为选择原则。一般采用轮盘赌的方式，也就是适应度值越高的个体越有机会繁殖后代，也有一些其他选择方法，例如联赛机制等。参与被选择的个体可以是上次迭代产生的子代，也可以是上次迭代的子代和父代。根据“适者生存”原则选择下一代的个体。如式（1.4）为选中 x_i 为下一代个体的次数，显然，

适应度较高的个体，繁殖下一代的数目较多；适应度较小的个体，繁殖下一代的数目较少，甚至被淘汰。这样，就产生了对环境适应能力较强的后代。对于问题求解角度来讲，就是选择出和最优解较接近的中间解。

$$P\{\text{选中 } x_i\} = \frac{f(x_i)}{\sum_{j=1}^{n} f(x_j)} \cdot n \tag{1.4}$$

交叉重组是由两个父代交换部分基因形成两个子代。随机地选择两个个体的相同位置，按交叉概率 P_c 在选中的位置实行交换。这个过程反映了随机信息交换，目的在于产生新的基因组合，即产生新的个体。一般采用单点等位基因交叉的方式，后来也出现了多点交叉、任意点交叉等方式。对于实数编码的染色体，则采用代数交叉，两个父代分别为 x_1 和 x_2，交叉后的子代分别为 $\lambda x_1 + (1-\lambda)x_2$ 和 $(1-\lambda)x_1 + \lambda x_2$，其中 $\lambda = \text{rand}(0,1)$。

变异是根据生物遗传中基因变异的原理，以变异概率 P_m 对某些个体的某些位执行变异。在变异时，对执行变异的串的对应位求反，即把 1 变为 0，把 0 变为 1。变异概率 P_m 与生物变异极小的情况一致，所以，P_m 的取值较小，一般取 0.01～0.2。

对于二进制编码的染色体，变异则是随机将某些位变反；实数编码染色体则采用在值上加正态噪声 $N(0,\delta)$ 的方法，也称为高斯变异，其中 δ 可随迭代的代数 t 的增长而递减，例如使用类似 $\delta = 1/(1+\sqrt{t})$ 的函数。

通过离散空间的 Markov 链可以证明，标准 GA 是无法保证全局收敛的[8,9]，但保存最佳个体（Elitism）的 GA 可以保证在时间无限条件下以概率 1 全局收敛的[10]。GA 的流程如下所示。

```
Begin
  初始化种群和参数;
  评价个体适应度（给出目标函数 f(x)，则 f(x_i) 称为个体 x_i 的适应度）。
  While（终止准则不满足）do
    选择（Selection）;
    交叉（Crossover）;
    变异（Mutation）;
    评价个体适应度。
  End While
  输出最优个体。
End
```

在上述步骤中初始化种群即创建一个随机的初始种群，个体记为 x_i（$i = 1,2\cdots,n$）。这个初始的群体也就是问题假设解的集合。将这些解比喻为染色体

或基因，该种群被称为第一代。问题的最优解将通过这些初始假设解进化而求出。通常算法的迭代终止准则选取以下两个。

（1）最优个体的适应度达到给定的阈值。

（2）算法的最大迭代次数到达（通常这时最优个体的适应度和群体适应度不再上升）。

由于 GA 是由进化论和遗传学机理而产生的搜索算法，所以在这个算法中会用到很多生物遗传学知识，下面对一些术语进行一下说明。

（1）染色体（Chromosome）。染色体又可以叫做基因型个体，一定数量的个体组成了群体（Population），群体中个体的数量叫做群体大小。

（2）基因（Gene）。基因是串中的元素，基因用于表示个体的特征。例如有一个串 $S=1011$，则其中的 1、0、1、1 这 4 个元素分别称为基因。它们的值称为等位基因。

（3）基因地点（Locus）。基因地点在算法中表示一个基因在串中的位置，也称为基因位置（Gene Position），有时也简称基因位。基因位置由串的左向右计算，例如在串 $S=1101$ 中，0 的基因位置是 3。

（4）基因特征值（Gene Feature）。在用串表示整数时，基因的特征值与二进制数的权一致，例如在串 $S=1011$ 中，基因位置 3 中的 1，它的基因特征值为 2；基因位置 1 中的 1，它的基因特征值为 8。

（5）适应度（Fitness）。各个个体对环境的适应程度叫做适应度。为了体现染色体的适应能力，引入了对问题中的每一个染色体都能进行度量的函数，叫适应度函数。这个函数是计算个体在群体中被使用的概率。

GA 是解决搜索问题的一种通用算法，对于各种通用问题都可以使用。同其他搜索算法相比，GA 还具有以下几方面的特点。

（1）GA 从问题解的串集开始搜索，而不是从单个解开始。这是 GA 与传统优化算法的极大区别。传统优化算法是从单个初始值迭代求最优解的，容易误入局部最优解。遗传算法从串集开始搜索，覆盖面大，利于全局择优。

（2）许多传统搜索算法都是单点搜索算法，容易陷入局部的最优解。GA 同时处理群体中的多个个体，即对搜索空间中的多个解进行评估，减少了陷入局部最优解的风险，同时算法本身易于实现并行化。

（3）GA 基本上不用搜索空间的知识或其他辅助信息，而仅用适应度函数值来评估个体，在此基础上进行遗传操作。适应度函数不仅不受连续可微的约束，而且其定义域可以任意设定。这一特点使得 GA 的应用范围大大扩展。

（4）GA 不是采用确定性规则，而是采用概率的变迁规则来指导其搜索方向。

（5）具有自组织、自适应和自学习性。GA 利用进化过程获得的信息自行组织搜索时，硬度大的个体具有较高的生存概率，并获得更适应环境的基因结构。

由于 GA 的整体搜索策略和优化搜索方法在计算时不依赖于梯度信息或其他辅助知识，而只需要影响搜索方向的目标函数和相应的适应度函数，所以 GA 提供了一种求解复杂系统问题的通用框架，它不依赖于问题的具体领域，对问题的种类有很强的鲁棒性，所以广泛应用于许多科学领域。

（1）函数优化。函数优化是遗传算法的经典应用领域，也是 GA 进行性能评价的常用算例，许多人构造出了各种各样复杂形式的测试函数：连续函数和离散函数、凸函数和凹函数、低维函数和高维函数、单峰函数和多峰函数等。对于一些非线性、多模型、多目标的函数优化问题，用其他优化方法较难求解，而 GA 可以方便地得到较好的结果。

（2）组合优化。随着问题规模的增大，组合优化问题的搜索空间也急剧增大，有时在目前的计算上用枚举法很难求出最优解。对这类复杂的问题，人们已经意识到应把主要精力放在寻求满意解上，而 GA 是寻求这种满意解的最佳工具之一。实践证明，GA 对于组合优化中的 NP 问题非常有效。例如 GA 已经在求解旅行商问题、背包问题、装箱问题、图形划分问题等方面得到成功的应用。

此外，GA 也在生产调度问题、自动控制、机器人学、图像处理、人工生命、遗传编码和机器学习等方面获得了广泛的运用。

1.2.3　进化规划

进化规划（Evolutionary Programming，EP）是由美国的 Fogel 于 20 世纪 60 年代提出来的，Fogel、Owens 和 Walsh 出版的《Artificial Intelligence Through Simulated Evolution》被认为是第一本关于 EP 的专著[11]。Fogel 在这部专著中系统阐述了进化规划的思想。他提出的方法与 GA 有许多共同之处，但不像 GA 那样注重父代与子代在遗传细节上的联系，而是把侧重点放在父代与子代表现行为的联系上。EP 仅使用变异与选择算子，而绝对不使用任何重组算子。其变异算子与进化策略的变异相类似，也是对父代个体采用基于正态分布的变异操作进行变异，生成相同数量的子代个体，即 m 个父代个体总共产生 m 个子代个体。EP 采用一种随机竞争选择方法，从父代和子代的并集中选择出 m 个个体构成下一代群体。其选择过程如下：对于由父代个体和子代个体组成的大小为 $2m$ 的临时群体中的每一个个体，从其他 $2m-1$ 个个体中随机等概率地选取出 q 个个体与其进行比较。在每次比较中，若该个体的适应度值不小于与之比较的个体的适应度值，则称该个体获得 1 次胜利。从 $2m$ 个个体中选择出获胜次数最多的 m 个个体作为下一代群体。

在 EP 中，假设问题的搜索空间是一个 n 维空间，则搜索点是一个 n 维向量：$\boldsymbol{X} \in \boldsymbol{R}^n$，进化群体的每个个体等同于这个 $\boldsymbol{X} \in \boldsymbol{R}^n$，所有的个体构成了整个群体。选择一个整数 N 作为群体的规模参数，随机生成搜索空间的 N 个初始个体作为初始群体。一般来说，这些个体的适应度值较差。EP 就是从这一初始群体出发，通过遗传操作，模拟进化过程，最后获得非常优秀的群体和个体。在 EP 中，个体适应度 $F(\boldsymbol{X})$ 是由它所对应的目标函数 $f(x)$ 通过某种比例变换而得到的。这种比例变换保证各个个体的适应度总取正值：

$$F(\boldsymbol{X}) = \delta[f(x)] \tag{1.5}$$

其中，δ 为某种比例变换函数。

在变异（Mutation）操作中，标准 EP 采用的是高斯变异算子。假设群体中某一个体 $\boldsymbol{X} = (x_1, x_2, \cdots, x_n)$，经过变异运算后得到一个新的个体 $\boldsymbol{X}' = (x_1', x_2', \cdots, x_n')$，则变异算子把个体 $\boldsymbol{X}$ 的每个分量 x_i 作用一个标准偏差得到新个体的组成元素。

$$x_i' = x_i + \delta_i N_i(0,1) \tag{1.6}$$

其中，$\delta_i = \beta_i \cdot f(x) + \gamma_i$，$i = 1, 2, \cdots, n$；$N_i(0,1)$ 表示对每个下标 i 都重新取值的均值为 0、方差为 1 的符合正态分布的随机变量；系数 β_i、γ_i 是特定的参数，一般取 $\beta_i = 1$，$\gamma_i = 0$。进化规划着眼于整个群体的进化，强调的是“物种的进化过程”。所以进化规划不使用交叉运算之类的个体重组算子，高斯变异算子的生物基础是强调个体的进化机制。

选择（Selection）体现了“适者生存”的自然法则。一般按照一种随机竞争的方式来进行（联赛选择方式），即在 N 个父辈个体 $P(t)$ 每个经过一次变异产生 N 个子代个体 $P'(t)$ 后，EP 利用一种随机 q 竞争选择方法从父辈和子代组成的共含有 $2N$ 个个体的个体集合 $\{P(t) \cup P'(t)\}$ 中选择 N 个个体，其中 $q \geqslant 1$ 是选择算子的参数。基本过程如下：对于每个个体 $\boldsymbol{X}_k \in \{P(t) \cup P'(t)\}$，从 $\{P(t) \cup P'(t)\}$ 中随机选取 q 个个体，比较这 q 个个体与个体 $\boldsymbol{X}_k$ 之间的适应度大小，以其中适应度比 $\boldsymbol{X}_k$ 的适应度还要差的个体的数目作为个体 $\boldsymbol{X}_k$ 的得分 $W_k (k = 1, 2, \cdots, 2N)$。在所有 $2N$ 个个体都经过了这种比较过程后，按得分 W_k 的大小做降序排列。选择前 N 个个体作为下一代群体 $P(t+1)$。

从上述选择算子可以知道，在进化过程中，每代群体中最好的个体在比较适应度大小时总被赋予了最大的得分，从而最好的个体能够确保被保留到下一代群体中。标准 EP（Classical Evolutionary Programming，CEP）的主要流程如下所示。

```
Begin
  产生初始群体;
```

```
  评价适应度（适应度函数是个体竞争的测度，控制个体生存的机会）。
  While（终止准则不满足）do
    变异（Mutation）;
    选择（Selection）;
    评价适应度。
  End While
  输出最优个体。
End
```

在 EP 中终止准则主要有以下几种。

（1）EP 已找到能接受的优秀个体。

（2）EP 已进化了预定的最大代数。

（3）在预定的代数内最适应个体的适应度无改进。

（4）最适应个体占群体的比例已达到规定的比例。

EP 的特点主要有以下几点。

（1）EP 不采用某种特定的编码结构，而直接对实际参变量（十进制）进行操作。

（2）EP 同时搜索解空间中的一群点，而非单点。EP 同时对空间中不同的区域采样，并构成不断进化的群体序列，或者说 EP 并行地爬多个山峰。这一特点使 EP 可以有效地防止搜索过程限于局部最优解，而具有较大的可能求得全局最优解。

（3）EP 对搜索空间没有任何特殊要求（如连续性等），对目标函数几乎无限制，不要求连续可微，既可以是显函数，也可以是映射矩阵甚至神经网络等隐函数，仅要求可用适应度函数评价个体。EP 的这一特点再一次拓宽了其应用领域。

（4）EP 采用概率变迁规则而非确定性规则来指导其搜索空间。EP 采用概率作为一种工具来启发搜索，有明确的搜索方向，比随机搜索方法有更高的搜索效率。

（5）EP 具有隐含并行性。不但使 EP 再次提高了搜索效率，而且易于采用并行机作并行高速运算，发展潜力很大。

EP 与 GA 作为进化算法，它们之间既有相同之处也有不同。相同之处有以下三点。

（1）操作一个候选解群体。

（2）限制一些候选解的被选择权

（3）确定一个选择准则得到下一代群体。

不同之处也有以下三点。

（1）个体的表示方法不同。GA 的处理对象不是参变量本身，而是参变量编码后的染色体串。因此存在基因型与表现型之间的映射问题。EP 不采用某种特定的编码结构，而直接对实际参变量（十进制）进行操作。

（2）遗传算子不同。GA 强调用交叉、翻转和变异的遗传算子应用到抽象的染色体；EP 强调父辈与子代之间的变异过程，重组只能应用到个体，而不能应用到整个群体。

（3）选择准则不同。在 GA 中每个父辈均有机会产生后代，而且产生后代的数量与其适应度值有关；在 EP 中每个父辈只产生一个后代，并且还要和后代一起竞争其生存权。

1.2.4　进化策略

虽然进化策略（Evolutionary Strategies，ES）的思想与 EP 的思想有很多相似之处，但它是在欧洲独立于 GA 和 EP 而发展起来的。1963 年，当时德国柏林工业大学的 ReehenbergL 和 Schwfel 等在进行风洞实验时，由于在设计中描述物体形状的参数难以用传统的方法进行优化，从而他们利用自然突变和自然选择的生物进化思想，对物体的外形参数进行随机变化并获得了较好的结果。随后，他们便对这种方法进行了深入的研究和发展，形成了进化计算的另一个分支，即进化策略（ES）[12]。早期 ES 的种群中只含有一个个体，且仅使用变异一种算子。在每一次进化过程中，通过变异后的子体与父体的比较来选择两者中的最优。这一策略称之为（1+1）策略。由于（1+l）策略存在效率较低，有时收敛不到全局最优解等不足，所以需要进行改进，其改进的方法就是增加种群内个体的数量，即后来的（μ+1）策略。此时种群内包含μ个个体，随机抽取其中的一个个体进行变异，然后更新群中的最差个体。为了进一步提高搜索的效率，随后又相继发展了（$\mu+\lambda$）策略和（μ，λ）策略。（$\mu+\lambda$）策略是根据种群内的μ个个体通过变异和重组操作产生 n 个个体，然后通过$\mu+\lambda$个个体的比较，从中选取μ个最优者；而（μ，λ）策略则是在新产生的λ（μ）个个体中通过比较选取μ个最优者。下面简单介绍 ES 的一般形式。

1．表示法和适应度值度量

在 ES 中，搜索点是 n 维向量 $x \in \boldsymbol{R}^n$，个体的适应度值等于其函数值 $\phi(a)=f(x)$。其中 x 是个体 a 的目标变量部分。此外，每个个体可以包括最多 n 个不同的方差 $c_n=\delta_i^2, i\in\{1,2,\cdots,n\}$ 和最多 $n\cdot(n-1)/2$ 个协方差 $c_{ij}=\delta_{ij}^2$，$i\in\{1,2,\cdots,n-1\}$，$j\in\{i+1,\cdots,n\}$。从而最多有 $w=n\cdot(n-1)/2$ 个策略参数可以和目标变量组合在一起构成一个个体 $a\in \boldsymbol{I}=\boldsymbol{R}^{n+w}$。不过，一般只考虑方差，从而 $a\in \boldsymbol{I}=\boldsymbol{R}^{2n}$，有时甚至对所有目标变量只用一个共同的方差，这时 $a\in \boldsymbol{I}=\boldsymbol{R}^{n+1}$。

2．变异

进化策略中，全体 $a=(x,\delta)$ 在变异算子作用下变为 $a'=(x',\delta')$，这里

$$\begin{aligned}\delta_i' &= \delta_i \cdot \mathrm{e}^{\tau' \cdot N(0,1)+\tau \cdot N_i(0,1)} \\ x_i' &= x_i + N(0,\delta_i'),\ \ i=1,2,\cdots,n\end{aligned} \tag{1.7}$$

其中，$N(0,1)$ 表示具有期望值为 0、标准偏差为 1 的正态分布随机变量，τ' 和 τ 是算子集参数，分别定义整体和个体步长。

3．重组

在进化策略中，重组算子可以按下列方式产生一个个体。

无重组：

$$x_t' = x_{S,t}$$

直接重组：

$$x_t' = x_{S,t} \quad 或 \quad x_{T,t}$$

加权平均重组：

$$x_t' = x_{S,t} + k(x_{T,i} - x_{S,t})$$

下标“S”和“T”指从 $P(t)$中随机选取的两个父辈个体，$k \in [0,1]$ 为一致随机变量。

4．选择

在进化策略中，选择是按完全确定的方式进行。$(\mu,\lambda)-ES$ 是从几个子代个体集中选择 $\mu(1 \leqslant \mu \leqslant \lambda)$个最好的个体；$(\mu+\lambda)-ES$ 是从父辈和子代个体的并集中选择μ 个最好的个体。虽然 $(\mu+\lambda)-ES$ 保留最优的个体能保证性能单调提高，但这种策略不能处理变化的环境，因此，目前选用最多的还是 $(\mu,\lambda)-ES$，研究表明比率 $\mu/\lambda \approx 1/T$ 是最优的。

由上面的讨论可知，进化策略算法的流程如下所示。

```
Begin
  确定问题的表达方式;
  随机生成初始群体;
  评价个体适应度。
```

```
While（终止准则不满足）do
    重组：将两个父代个体交换目标变量和随机因子，产生新个体；
    突变：对重组后的个体添加随机量，产生新个体；
    计算新个体适应度；
    选择：根据选择策略，挑选优良个体组成下一代群体。
End While
输出最优个体。
End
```

GA 与 ES 都是模拟生物界自然进化过程而建立的鲁棒性计算机算法，同属于进化计算。在统一框架下对二者进行比较，可以发现它们有许多相似之处，例如它们的算法中都存在变异算子。GA 与 ES 的不同之处有以下几点。

（1）应用领域不同。GA 从广义上说是一种自适应搜索技术，应用于参数优化、机器学习、免疫系统等许多领域；ES 是一种数值优化的方法，主要应用于离散型优化问题。

（2）求解空间操作不同。GA 是将原问题的解空间映射到串空间之中，然后再施行遗传操作，它强调个体基因结构的变化对其适应度的影响。而 ES 直接在解空间上进行操作，强调进化过程中从父体到后代行为的自适应性和多样性，强调进化过程中搜索步长的自适应性调节。

（3）ES 和 GA 表示个体的方式不同。ES 在浮点矢量上运行，而 GA 一般运行在二进制矢量上。

（4）ES 和 GA 的选择过程不同。标准 GA 对每一个个体都指定一个非零的选择概率；ES 确定地把某些个体排除在被选择之外。

（5）ES 和 GA 的复制参数不同。GA 的复制参数（交叉和变异的可能性）在进化过程中保持恒定，而 ES 时时改变它们。ES 中变异作为主要搜索算子，而标准 GA 中，变异只处于次要位置。

1.2.5　差分进化算法

差分进化（Differential Evolution，DE）算法是一种新兴的进化计算技术，或称为差分演化算法、微分进化算法、微分演化算法、差异演化算法。它是由 Storn 等人于 1995 年提出的，最初的设想是用于解决切比雪夫多项式问题，后来发现 DE 算法也是解决复杂优化问题的有效技术[13]。DE 算法与人工生命，特别是进化算法有着极为特殊的联系。DE 算法是基于群体智能理论的优化算法，通过群体内个体间的合

作与竞争产生的群体智能指导优化搜索。但相比于进化算法，DE 算法保留了基于种群的全局搜索策略，采用实数编码基于差分的简单变异操作和一对一的竞争生存策略，降低了遗传操作的复杂性。同时，DE 算法特有的记忆能力使其可以动态跟踪当前的搜索情况，以调整其搜索策略，具有较强的全局收敛能力和鲁棒性，且不需要借助问题的特征信息，适于求解一些利用常规的数学规划方法所无法求解的复杂环境中的优化问题。

DE 算法是一种基于群体进化的算法，具有记忆个体最优解和种群内信息共享的特点，即通过种群内个体间的合作与竞争来实现对优化问题的求解，其本质是一种基于实数编码的具有保优思想的贪婪遗传算法。DE 算法首先在问题的可行解空间随机初始化种群，对当前种群进行变异和交叉操作产生另一个新种群，然后利用基于贪婪思想的选择操作对这两个种群进行一对一的选择，从而产生最终的新一代种群。DE 算法通过利用种群中个体的距离和方向信息来寻找全局最优解。为此，种群中的每个个体通过动态改变差分项的方向和步长来调整空间的搜索行为。在每一代中，变异和交叉操作均作用于个体，以此产生一个新的种群。然后利用选择操作在这新旧种群中进行比较，来选择产生下一代种群。标准 DE 算法流程流程如下所示。

Begin

随机初始化 DE 的种群;

根据适应度函数，评价个体;

确定具有最好目标值的 X_{best}。

While（终止准则不满足）do

对每个个体执行变异操作（Mutation），以获得相应的变异个体:

$$V_i(t+1) = X_{\text{best}}(t) + F[X_{r1}(t) - X_{r2}(t)]$$

对个体和其变异个体执行交叉操作（Crossover），以获得试验个体;

$$u_{i,j}(t+1) = \begin{cases} v_{i,j}(t+1) & \text{若 rand}(j) < C_R \text{ 或 } j=\text{rand}(i)，C_R \text{ 为交叉因子} \\ x_{i,j}(t) & \text{其他; } j=1,2,\cdots,d \end{cases}$$

根据适应度函数，评价试验个体的目标值;

在个体和其试验个体之间进行选择操作（Selection），以生成新个体;

$$X_i(t+1) = \begin{cases} U_i(t+1) & \text{若} F[U_i(t+1)] < F[X_i(t)] \\ X_i(t) & \text{其他} \end{cases}$$

```
    更新 X_best。
  End While
  输出 X_best 及其目标值。
End
```

DE 算法的搜索性能取决于算法全局探索和局部开发能力的平衡，而这在很大程度上依赖于算法的控制参数的选取，包括种群规模缩放比例因子和交叉因子等。相对其他进化算法而言，DE 算法所需调节的控制参数较少，合理的参数设置可以参考相关文献。归纳起来，DE 算法具有如下优点。

（1）算法通用，不依赖于问题信息。

（2）算法原理简单，容易实现。

（3）群体搜索，具有记忆个体最优解的能力。

（4）协同搜索，具有利用个体局部信息和群体全局信息指导算法进一步搜索的能力。

（5）易于与其他算法混合，构造出具有更优性能的算法。

DE 算法性能优越、容易理解、易于实现，一经提出就备受关注并得到了广泛的应用，但是 DE 算法本身仍有很多值得研究的地方，例如参数的设置问题。DE 算法的性能很大程度上和参数的选取有关，比较重要的有群体规模、缩放因子和交叉因子。

1．群体规模的选择

从计算复杂度分析，种群规模越大，搜索到全局最优解的可能性就越大，然而所需的计算量和计算时间也要增加。而且最优解的质量并非随种群规模的增大一味变好，有时种群规模的增大，反而会使最优解的精度降低，因此，合理选取种群规模对算法的搜索效率的提高具有重要意义。

当种群规模增大到一定个数时，解的精度不再提高，甚至会出现降低的情况。这是因为较大的种群规模能保持种群的多样性，但会降低收敛速度，多样性和收敛速度必须保持一定的平衡。因此，当种群规模太大时，如果不增加最大进化代数，精度反而会降低。另外，种群规模越大，多样性就越大，所以如果种群过早收敛，就要增加种群规模以增加多样性。

在给定最大进化代数情况下，对低维简单问题，种群规模在[15，35]之间较好；对高维复杂问题，种群规模在[30，50]之间时，优化结果较好。总之，在给

定最大进化代数下，种群规模在[15，50]之间时，能很好地保持多样性和收敛速度的平衡。

2．缩放因子 F 对优化效果的影响

缩放因子 F 取值[0.5，1]之间时，算法得到的结果较好；当 $F<0.5$ 或 $F>1$ 时，算法求得的解的质量不高。缩放因子 F 是用于控制差分向量对变异个体 V 的影响的。当 F 较大时，差分向量对 V 的影响较大，能产生较大的扰动，从而有利于保持种群的多样性；反之，F 较小时，扰动较小，缩放因子能起到局部精细化搜索的作用。因此，F 对种群的多样性起到了一定的调节作用。缩放因子 F 取值太大，虽然能保持种群多样性，但算法近似随机搜索，搜索效率低下，求得的全局最优解精度低；反之，F 太小，种群多样性丧失很快，算法易于陷入局部最优出现早熟收敛。

由于 DE 算法是一种“贪婪”选择算法，所以，随着种群的不断进化，各个个体逐渐靠近最优个体，个体间的差异也会逐渐减小。这就意味着，当算法进化到一定程度时，种群多样性就会丧失。种群多样性对算法的全局搜索能力有一定影响。种群多样性大，增加了从局部最优逃脱的可能性，有利于全局搜索，但会降低收敛速度；种群多样性小，有利于局部搜索，收敛速度快，但是，易于陷入局部最优，出现所谓的早熟现象。

综上，F 对算法的局部搜索和全局搜索起到了一定的调节作用。F 较大，有利于保持种群多样性和全局搜索；F 较小，有利于局部搜索和提高收敛速度。

3．交叉因子 C_R 对优化效果的影响

C_R 的值较小时，所需的函数评价次数较大，收敛速度较慢，但成功率较高，算法的稳定性好；C_R 的值较大时，常常会加速收敛，但易于陷入局部最优，发生早熟现象，达到给定精度的成功率低，稳定性差。可见，成功率和收敛速度是一对矛盾。因此，为了同时保证较高的成功率和较快的收敛速度，对于单峰函数，C_R 取值相对较大些，在[0.6，0.8]区间；对其他复杂、多峰函数 C_R 取值应相对小些，在[0.1，0.5]区间。

新子代个体 U 是由变异个体 V 和父代个体 X 分量间相互交叉而产生的。C_R 的值越大，则 V 对 U 的贡献越多，有利于开拓新空间，从而加速收敛，但在后期变异个体趋于同一，不利于保持多样性，从而易于“早熟”，稳定性差；C_R 的值越小，则 X 对 U 的贡献越多，这样就减弱了算法开拓新空间的能力，收敛速度相对较慢，但有利于保持种群多样性，从而能有较高的成功率。

1.3　群体智能

群体智能（Swarm Intelligence，SI）就是无智能或具有简单智能的个体在无集中控制的情况下，通过单个个体自身的简单行为，使得整个群体表现出某种智能行为，从而解决某些特定的问题。

1.3.1　群体智能概述

SI 的概念最早由 Beni、Hackwood 和 Wang 在分子自动机系统中提出的[14,15]。分子自动机中的主体在一维或二维网格空间中与相邻个体相互作用，从而实现自组织。1999 年，Bonabeau、Dorigo 和 Theraulaz 在他们的著作《Swarm Intelligence: From Natural to Artificial Systems》中对 SI 进行了详细的论述和分析，给出了 SI 的一种不严格定义：任何一种由昆虫群体或其他动物社会行为机制而激发设计出的算法或分布式解决问题的策略均属于 SI[16]。

这里，群体（Swarm）可被描述为一些相互作用相邻个体的集合体，鱼群、蜂群、蚁群、鸟群都是群体的典型例子。鱼聚集成群可以有效地逃避捕食者，因为任何一只鱼发现异常都可带动整个鱼群逃避。蚂蚁成群则有利于寻找食物，因为任一只蚂蚁发现食物都可带领蚁群来共同搬运和进食。一只蜜蜂或蚂蚁的行为能力非常有限，它几乎不可能独立存在于自然世界中，而多个蜜蜂或蚂蚁形成的群体则具有非常强的生存能力，且这种能力不是多个个体之间能力通过简单叠加所获得的。社会性动物群体所拥有的这种特性能帮助个体很好地适应环境，个体所能获得的信息远比它通过自身感觉器官所取得的多，其根本原因在于个体之间存在着信息交互能力。信息的交互过程不仅仅在群体内传播了信息，而且群内个体还能处理信息，并根据所获得的信息（包括环境信息和附近其他个体的信息）改变自身的一些行为模式和规范，这样就使得群体涌现出一些单个个体所不具备的能力和特性，尤其是对环境的适应能力。这种对环境变化所具有适应的能力可以被认为是一种智能（关于适应性与智能之间的关系存在着一些争议，Fogel 认为智能就是具备适应的能力），也就是说动物个体通过聚集成群而涌现出了智能。因此，Bonabeau 将 SI 的定义进一步推广为：无智能或简单智能的主体通过任何形式的聚集协同而表现出智能行为的特性。这里我们关心的不是个体之间的竞争，而是它们之间的协同。

James Kennedy 和 Russell C.Eberhart 在 2001 年出版了《Swarm Intelligence》，是群智能发展的一个重要里程碑，因为此时已有一些群智能理论和方法得到了应用。他们不反对 Bonabeau 关于 SI 的定义，赞同其定义的基本精神，但反对定义中使用

“主体”一词[17]。其理由是“主体”所带有自治性和特殊性是许多群体的个体所不具备和拥有的，这将大大限制群体的定义范围。他们认为暂时无法给出合适的定义，赞同由 Mark Millonas 提出的构建一个 SI 系统所应满足的五条基本原则[18]。

（1）Proximity Principle：群内个体具有能执行简单的时间或空间上的评估和计算的能力。

（2）Quality Principle：群内个体能对环境（包括群内其他个体）的关键性因素的变化做出响应。

（3）Principle of Diverse Response：群内不同个体对环境中的某一变化所表现出的响应行为具有多样性。

（4）Stability Principle：不是每次环境的变化都会导致整个群体的行为模式的改变。

（5）Adaptability Principle：环境所发生的变化中，若出现群体值得付出代价的改变机遇，群体必须能够改变其行为模式。

《Swarm Intelligence》最重要的观点是“Mind is Social”[17]，也就是认为人的智能是源于社会性的相互作用，文化和认知是人类社会性不可分割的重要部分，这一观点成了群体智能发展的基石。SI 已成为有别于传统人工智能中连接主义、行为主义和符号主义的一种新的关于智能的描述方法。SI 的思路为在没有集中控制且不提供全局模型的前提下寻找复杂的分布式问题求解方案提供了基础。在计算智能领域已取得成功的两种基于 SI 的优化算法是蚁群优化算法和粒子群优化算法。目前，已有的基于 SI 的优化算法都是源于对动物社会通过协作解决问题行为的模拟，它主要强调对社会系统中个体之间相互协同作用的模拟。这一点与 EC 不同，EC 是对生物演化中适者生存的模拟。

与 EC 一样的是，SI 的目的并不是为了忠实地模拟自然现象，而是利用他们的某些特点去解决实际问题。另一个与 EC 的相同点是，基于 SI 的优化算法也是概率搜索算法。目前，已有的 SI 理论和应用研究证明 SI 方法是一种能够有效解决大多数优化问题的新方法，更重要的是，SI 潜在的并行性和分布式特点为处理大量的以数据库形式存在的数据提供了技术保证。

无论是从理论研究还是应用研究的角度分析，SI 理论及应用研究都是具有重要学术意义和现实价值的。由于 SI 理论依据是源于对生物群落社会性的模拟，因此其相关数学分析还比较薄弱，这就导致了现有研究还存在一些问题[19]。首先，SI 算法的数学理论基础相对薄弱，缺乏具备普遍意义的理论性分析，算法中涉及的各种参数设置一直没有确切的理论依据，通常都是按照经验方法确定，对具体问题和应用环境的依赖性比较大。其次，同其他的自适应问题处理方法一样，SI 也不具备绝对

的可信性，当处理突发事件时,系统的反应可能是不可测的，这在一定程度上增加了其应用风险。

1.3.2　蚁群优化算法

蚁群优化（Ant Colony Optimization，ACO）算法由 Colorni、Dorigo 和 Maniezzo 于 1991 年提出的[20,21]，它是通过模拟自然界蚂蚁社会寻找食物的方式而得出的一种仿生优化算法。自然界中蚁群寻找食物时会派出一些蚂蚁分头在四周游荡，如果一只蚂蚁找到食物，它就返回巢中通知同伴并沿途留下“信息素”（Pheromone）作为蚁群前往食物所在地的标记。信息素会逐渐挥发，如果两只蚂蚁同时找到同一食物，又采取不同路线回到巢中，那么比较绕弯的一条路上信息素的气味会比较淡，蚁群将倾向于沿另一条更近的路线前往食物所在地。ACO 算法设计虚拟的“蚂蚁”，让它们摸索不同路线，并留下会随时间逐渐消失的虚拟“信息素”。根据“信息素较浓的路线更近”的原则，即可选择出最佳路线。

ACO 算法可以看成是一种基于解空间参数化概率分布模型（Parameterized Probabilistic Model）的搜索算法框架，其中解空间参数化概率模型的参数就是信息素，因而这种模型就是信息素模型。在基于模型的搜索算法框架中，可行解通过在一个解空间参数化概率分布模型上的搜索产生，此模型的参数用以前产生的解来进行更新，使得在新模型上的搜索能集中在高质量的解搜索空间内。这种方法的有效性建立在高质量的解总是包含好的解构成元素的假设前提下。通过学习这些解构成元素对解的质量的影响有助于找到一种机制，并通过解构成元素的最佳组合来构造出高质量的解。一般来说，一个基于模型的搜索算法通常使用以下两步迭代来解决优化问题。

（1）可行解通过在解空间参数化概率分布模型上的搜索产生。

（2）用搜索产生的解来更新参数化概率模型，即更新解空间参数化概率分布的参数，使得在新模型上的搜索能集中在高质量的解搜索空间内。

在 ACO 算法中，基于信息素的解空间参数化概率模型（信息素模型）以解构造图的形式给出。在解构造图上，定义了一种作为随机搜索机制的人工蚂蚁，蚂蚁通过一种分布在解构造图上被称为信息素的局部信息的指引，在解构造图上移动，从而逐步地构造出问题的可行解。信息素与解构造图上的节点或弧相关联，作为解空间参数化概率分布模型的参数。由于旅行商问题（Traveling Salesman Problem，TSP）能够非常直接地映射为解构造图（城市为节点，城市间的路径为弧，信息素分布在弧上），且是一个 NP 难题，因而 ACO 算法的大部分成果都集中在 TSP 问题上。一般而言，用于求解 TSP 问题、生产调度问题等组合优化问题的 ACO 算法都遵循如下的统一算法流程。

```
Begin
  设置参数，初始化信息素踪迹。
  While（不满足终止准则）do
    For 蚁群中的每只蚂蚁
      For 每个解构造步（直到构造出完整的可行解）
        蚂蚁按照信息素及启发式信息的指引构造一步问题的解;
        进行信息素局部更新。（可选）
      End For
    End For
    （1）以某些已获得的解为起点进行邻域（局部）搜索;（可选）
    （2）根据某些已获得的解的质量进行全局信息素更新。
  End While
End
```

在算法中，蚂蚁逐步构造问题的可行解，在一步解的构造过程中，蚂蚁以概率方式选择信息素强且启发因子高的弧到达下一个节点，直到不能继续移动为止。此时蚂蚁所走过的路径对应求解问题的一个可行解。局部信息素更新针对蚂蚁当前走过的一步路径上的信息素进行，全局信息素更新是在所有蚂蚁找到可行解之后，根据发现解的质量或当前算法找到的最好解对路径上的信息素进行更新。总的来说，ACO 算法的主要特点可以概括为以下几点。

（l）采用分布式控制，不存在中心控制。

（2）每个个体只能感知局部的信息，不能直接使用全局信息。

（3）个体可以改变环境，并通过环境来进行间接通信。

（4）具有自组织性，即群体的复杂行为是通过个体的交互过程中突现出来的智能。

（5）是一类概率型的全局搜索方法，这种非确定性是算法能够有更多的机会求得全局最优解。

（6）其优化过程不依赖于优化问题本身的严格数学性质，比如连续性、可导性以及目标函数和约束函数的精确数学描述。

（7）是一种基于多主体（Multi-Agent）的智能算法，各主体之间通过相互协作来更好的适应环境。

（8）具有潜在的并行性，其搜索过程不是从一点出发，而是同时从多个点同时进行。这种分布式多智能体的协作过程是异步并发进行的，分布并行模式将大大提高整个算法的运行效率和快速反应能力。

1.3.3　粒子群优化算法

粒子群优化（Particle Swarm Optimization，PSO）算法是模拟鸟类捕食行为的群体智能算法，由美国电气工程师 Eberhart 和社会心理学家 Kennedy 于 1995 年提出[22]。由于 PSO 算法容易实现，需要调整的参数少，一经提出就受到了研究者的重视，被广泛应用到各个领域。有关 PSO 算法的概念、原理和应用将在后续章节详细介绍。

1.3.4　其他智能算法

人工鱼群算法也是一个模拟自然界生物特性的智能算法，它是由浙江大学的李晓磊[23]等人在 2002 年首次提出以来，并得到了国内外学者的广泛关注。目前对该算法的研究应用已经渗透到多个应用领域，由最初的求解一维静态优化问题已发展到解决多维动态组合优化问题。在李晓磊的博士论文中，给出了人工鱼群算法的原理和算法步骤的详细描述，并对算法的收敛性能和算法中控制参数对算法性能的影响进行了分析。针对组合优化问题，提出了人工鱼群算法中的距离、邻域和中心等概念，并给出了算法在组合优化问题应用中的描述；针对大规模系统的优化问题，给出了基于分解协调思想的人工鱼群算法，并介绍了人工鱼群算法中常用的一些改进方法，以及人工鱼群算法在时变系统的在线辨识和 PID 参数整定中的应用分析；最后指出了鱼群模式和算法的今后研究发展方向。目前，人工鱼群算法已经成为交叉学科中一个非常活跃的前沿性研究问题。

人口迁移算法（Population Migration Algorithm，PMA）是我国学者周永华和毛宗源于 2003 年提出的一类模拟人口迁移机理的全局优化算法[24]，不仅具有通用、简单、并行、稳定等优点，而且还具有较强的全局搜索能力。与以往的模拟进化算法不同，人口迁移算法是建立在对人口移动规律的整体认识的基础上，对人口迁移的简单模拟。算法思想来源于社会领域中人口随经济重心而转移、随人口压力增加而扩散的规律，即模拟的是人往高处走，人往富处流，当某个优惠地区的相对人口压力增加时，人们就会迁出该优惠地区去寻找更好更适合自己的优惠地区的这种规律，这为人们进行算法设计提供了一种新的思路与方向。已有数值实验和应用说明人口迁移算法具有良好的全局搜索能力，但理论基础以及广泛的实际应用仍有待深入研究。

人工蜂群算法（Artificial Bee Colony Algorithm，ABCA）是 Karaboga 于 2005 年提出的一种新颖的 SI 优化算法[25]，主要模拟蜂群的智能采蜜行为，蜜蜂根据各自的分工进行不同的采蜜活动并实现蜜源信息的共享和交流从而找到问题的最优解。

在 ABCA 中，人工蜂群包含 3 个组成部分：采蜜蜂、观察蜂和侦察蜂。群体的一半由采蜜蜂构成，另一半由观察蜂构成；每一处蜜源仅仅有一个采蜜蜂，也就是说蜜源数和采蜜蜂的数目相等；放弃所采蜜源的采蜜蜂成为侦察蜂。搜索活动可以概括如下：采蜜蜂根据它们记忆中的蜜源位置在其邻域内确定另一个蜜源，并在蜂巢内将它们的信息通过舞蹈共享给观察蜂，观察蜂选择其中的一个蜜源；观察蜂根据所选择的蜜源在其邻域内搜索另一个蜜源；放弃所采蜜源的采蜜蜂将成为侦察蜂，并搜索一个新的随机蜜源。

在 ABCA 算法中每个蜜源的位置代表优化问题的一个可能解，蜜源的花蜜量对应于相应解的质量或适应度。ABCA 算法首先随机产生初始群体，即 n 个初始解，n 为采蜜蜂数也等于蜜源数目。每个解 $\boldsymbol{X}_i$（$i=1,2,\cdots,n$）是一个 d 维的向量。以这些初始解为基础采蜜蜂、观察蜂和侦察蜂开始进行循环搜索。采蜜蜂根据它记忆中的局部信息产生一个变化的位置，并检查新位置的花蜜量。如果新位置优于原位置，则该蜜蜂记住新位置并忘记原位置。所有的采蜜蜂完成搜索过程后，它们将所知道的蜜源信息通过舞蹈与观察蜂共享，观察蜂根据从采蜜蜂处得到的信息，按照与花蜜量相关的概率选择一个蜜源位置，并像采蜜蜂那样对记忆中的位置做一定的改变，并检查新候选位置的花蜜量。若新位置优于记忆中的位置，则用新位置替换原来的蜜源位置，否则保留原位置。换句话说“贪婪选择”机制被用于选择原位置和新的候选位置。

参 考 文 献

[1] Reeves C.R. Modern Heuristic Techniques for Combinatorial Problems. Oxford: Blackwell Scientific Publications, 1993.

[2] Wolpert D.H, Macready W.G. No Free Lunch Theorems for Search. Technical Report SFI-TR -95-02-010, Santa Fe Institure.1995:1～32.

[3] Wolpert D.H, Macready W.G.No Free Lunch Theorems for Optimization. IEEE TRANSACTION ON EVOLUTIONARY COMPUTATION,1997,1(1):67～82.

[4] Christensen S, Oppacher F. What Can We Learn from No Free Lunch?A First Attempt to Characterize the Concept of Searchable Function.The 2001 Genetic and Computation Conference. Piscataway, NJ:IEEE Press, 2001(12):19～1226.

[5] Fogel D.B. Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. NewYork: Wiley-IEEE Press,1995.

[6] Holland J.H. Adaptation in Natural and Artificial Systems. Ann Arbor:University

of Michigan.

[7] Goldberg D.E. Genetic algorithms in Search, Optimization, and Machine Learning. Reading MA: Addison-Wesley,1989.

[8] Goldberg D.E, Segrest P. Finite Markov Chain Analysis of Genetic Algorithms.In: The Internationa Conference on Genetic algorithms.Piscataway, NJ:IEEE Service Center,1987.1～8.

[9] Cerf R.Asymptotic Convergence of Genetic Algorithms.ADVANCES IN APPLIED PROBABILITY, 1998, 30(2):521～550.

[10] Leung Yee, Gao Yong,Xu Zong-Ben. A Markov Chain Analysis of Premature Convergence in Genetic Algorithms. In:The International Conference on Neural Information Processing, Springer Verlag, Vol(2),1996.1330～1334.

[11] Fogel L, Owens A and Walsh M. Artificial Intelligence through Simulated Evolution.New York:John Wiley,1966.

[12] Rechenberg I.Bionic. Evolution and Optimizing.Naturwissen Schaftliche Rundschau, 26(11): 465～472.

[13] Storn R and Price K.Differential Evolution-A Simple and Efficient Heuristic for Global Optimization Over Continuous Spaces, Journal of Global Optimization,11(4), 341～359.

[14] Hackwood S,Wang J.The Engineering of Cellular Robotic Systems.In:IEEE International Symposium on Intelligent Control.Piscataway,NJ:IEEE Press,1988.70～75.

[15] Hackwood S,Beni G.Self-organization of Sensors for Swarm Intelligence.In:IEEE International conference on Robotics and Automation.Piscataway,NJ:IEEE Press, 1992.819～829.

[16] E.Bonabeau, M.Dorigo, and G.Theraulaz.Swarm Intelligence:From Natural to Artificial Systems. New York: Oxford University Press, 1999.

[17] Kennedy James F, Eberhart Russell and Shi Yuhui. Swarm Intelligence, Elsevier Science Ltd, 2005.

[18] Millonas M.M. Swarms,Phase Transitions and Collective Intelligence. In: Langton, C.G.(Ed.), Artificial Life III.Santa Fe Institute Studies in the Sciences of Complexity,Vol.XVII. Addison- Wesley,1994:417～445.

[19] Mark Fleischer.Foundations of Swarm Intelligence:From Principles to Practice. In:Conference on Swarming:Network Enabled C4ISR.E-Print Alert Service.2003.1～13.

[20] Colorni A, Dorigo M and Maniezzo V.Distributed Optimization by Ant Colonies. The First European conference on Artificial Life.France:Elsevier,1991.134～142.

[21] Dorigo M, Maniezzo V and Colorni A.The Ant System: Optimization by a Colony of Cooperating Agents.IEEE TRANSACTIONS ON SYSTEM,MAN,AND CYBER NETICS—Part B.1996,26:29～41.

[22] Kennedy J,Eberhart R.C.Particle Swarm Optimization.In:IEEE International Conference on Neural Networks,IV.Piscataway,NJ:IEEE Service Center，1995.1942～1948.

[23] 李晓磊，邵之江，钱积新．一种基于动物自治体的寻优模式：鱼群算法．系统工程理论与实践，2002，22(11)：32～38.

[24] 周永华，毛宗源一种新的全局优化搜索算法一人口迁移算法．华南理工大学学报（自然科学版），31（3），2003（3）：l～5.

[25] D.Karaboga, An Idea Based On Honey Bee Swarm For Numerical Optimization, Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

第 2 章　基本PSO算法

粒子群优化（Particle Swarm Optimization，PSO）算法是一种优化计算技术，由 Eberhart 博士和 Kennedy 博士提出，源于对鸟群觅食行为的研究[1]。PSO 算法同遗传算法（GA）类似，是一种基于迭代的优化工具，系统初始化为一组随机解，通过迭代搜寻最优解。但是并没有 GA 用的交叉（Crossover）及变异（Mutation），而是粒子在解空间中追随最优的粒子进行搜索。同 GA 相比，PSO 算法的优势在于简单容易实现，能够记忆个体最优和全局最优信息，并且没有太多的参数需要调整，目前已广泛应用于函数优化、神经网络训练、模糊系统控制以及其他 GA 的应用领域。

2.1　PSO算法产生的背景

PSO 算法的产生主要源于 1994 年由 Holland 教授正式提出的复杂适应系统理论（Complex Adaptive System，CAS）以及人工生命的研究[2]。在 CAS 中的成员称为具有适应性的主体(Adaptive Agent)，简称为主体，比如研究鸟群系统，每个鸟在这个系统中就称为主体。主体的适应性，是指它能够与环境以及其他主体进行交流，在这种交流的过程中“学习”或“积累经验”，并且根据学到的经验改变自身的结构和行为方式。整个系统的演变或进化包括：新层次的产生（如小鸟的出生）；分化和多样性的出现（如鸟群中的鸟分成许多小的群）；新的主题的出现（如鸟寻找食物过程中，不断发现新的食物）。所以 CAS 系统中的主体具有 4 个基本特点（这些特点是 PSO 算法发展变化的依据）。

（1）主体是主动的、活动的。这点是 CAS 和其他建模方法的关键性的区别。具有适应性的主体的概念把个体主动性提高到了系统进化基本动因的位置，从而成为研究与考察宏观行为的出发点。

（2）主体与环境及其他主体相互影响、相互作用，这种影响是系统发展变化的主要动力。相互作用是“可记忆”的，它表现为进化过程中每个个体的结构和行为方式的变化，以不同的方式“存储”在个体内部。

（3）环境的影响是宏观的，主体之间的影响是微观的，宏观与微观要有机结合。

（4）整个系统可能还要受一些随机因素的影响。随机因素的影响不仅影响状态，而且影响组织结构和行为方式。具有主动性的个体会接受教训、总结经验，并且以某种方式把“经历”记住，使之“固化”在自己以后的行为方式中。

正因如此，CAS 理论提供了模拟生物、生态、经济、社会等复杂系统的巨大潜力。PSO 算法就是受到一个 CAS 系统——鸟群系统行为特性的启发建立的一种算法理论，源于 1987 年 Reynolds 对鸟群社会系统 Boids（Reynolds 对他仿真的鸟群系统的命名）的仿真研究[3]，因此 PSO 算法中也包含了 CAS 的四个基本特点。在 Boids 中，一群鸟在空中飞行，每个鸟遵守以下三条规则。

（1）避免与相邻的鸟发生碰撞冲突。

（2）尽量与自己周围的鸟在速度上保持协调和一致。

（3）尽量试图向自己所认为的群体中靠近。

不过 Reynolds 仅仅将其作为 CAS 的一个实例作仿真研究，而并未将它用于优化计算中，也没有任何实用价值。将其用于优化计算中的创造性工作是由 Kennedy 和 Eberhart 在 1995 年完成的，因此学术界均将 Kennedy 和 Eberhart 作为 PSO 的创始人。

人工生命（Artificial Life，AL）是用来研究具有某些生命基本特征的人工系统的[4]，其许多早期研究工作也起源于人工智能。在 1987 年第一次人工生命研讨会上，美国圣塔菲研究所（Santa Fe Institute，SFI）非线性研究组的 Langton 正式提出人工生命的概念，建立起人工生命新学科。随后，Holland 提出的人工生命定义为：人工生命是研究能够演示出自然生命系统特征行为的人造系统。近年来，人工生命的研究发展非常快，在某些方面的研究已与传统的生物科学形成了互补。人工生命包括两方面的内容。

（1）研究如何利用计算技术研究生物现象。

（2）研究如何利用生物技术研究计算问题。

在第（2）方面内容的研究中，现已经有了很多源于生物现象的计算技巧，例如人工神经网络是简化的大脑模型，遗传算法（GA）是模拟基因进化的过程。目前这一类计算技术被统称为自然计算（Nature Computation）。群体智能（SI）属于自然计算中的一类，它模拟另一种生物系统：社会系统。更确切地说，是模拟由简单个体组成的群落与环境以及个体之间的互动行为，这些模拟系统利用局部信息从而可能产生不可预测的群体行为。目前，在计算智能（Computational Intelligence）领域涌现了多种基于 SI 的优化算法，已得到广泛应用的有前面章节介绍的蚁群优化（Ant Colony Optimization，ACO）算法和本章将要介绍的 PSO 算法。ACO 算法是对蚂蚁群落食物采集过程的模拟，已经成功运用在很多路径优

化问题上，PSO 算法则是源于鸟群觅食过程的模拟，实践证明 PSO 算法是一种很好的优化工具。

Kennedy 和 Eberhart 在 Reynolds 的鸟群社会系统仿真系统 Boids 中加入了一个特定点，定义为食物，鸟根据周围鸟的觅食行为来寻找食物。他们的初衷是希望通过这种模型来模拟鸟群寻找食源的现象，然而实验结果却揭示这个仿真模型中蕴涵着很强的优化能力，尤其是在多维空间寻优中。由于在最初的仿真系统中每个鸟在计算机屏幕上表示为一个点（Points），而“点”这个词在数学领域具有非常多的意义，因此作者用了“粒子”（Particle）来表示每一个个体，于是也就得到了基本 PSO 算法。PSO 算法兼有进化计算和 SI 的特点，与其他进化算法相类似，PSO 算法也是通过个体间的协作与竞争，实现复杂空间中最优解的搜索。

2.2　基本PSO算法更新过程

如前所述，基本 PSO 算法模拟鸟群的觅食行为。设想这样一个场景：一群鸟在随机搜寻食物，在这个区域里只有一块食物，所有的鸟都不知道食物在哪里，但是它们知道当前的位置离食物还有多远。那么找到食物的最优策略是什么呢?最简单有效的就是搜寻目前离食物最近的鸟的周围区域。基本 PSO 算法从这种模型中得到启示并用于解决优化问题。在基本 PSO 算法中，每个优化问题都是搜索空间中的一只鸟，我们称之为“粒子”。所有的粒子都有一个被优化的函数决定的适应度值（Fitness Value），每个粒子还有一个速度决定它们飞翔的方向和距离，粒子追随当前的最优粒子在解空间中搜索。Reynolds 对鸟群飞行的研究发现，鸟仅仅是追踪它有限数量的邻居但最终的整体结果是整个鸟群好像在一个中心的控制之下，即复杂的全局行为是由简单规则的相互作用引起的。

可以这样来理解基本 PSO 算法：基本 PSO 算法就是模拟一群鸟寻找食物的过程，每个鸟就是基本 PSO 算法中的粒子，也就是我们需要求解问题的可能解，这些鸟在寻找食物的过程中，不停改变自己在空中飞行的位置与速度。大家也可以观察一下，鸟群在寻找食物的过程中，开始鸟群比较分散，逐渐这些鸟就会聚成一群，这个群忽高忽低、忽左忽右，直到最后找到食物。

基本 PSO 算法是种基于 SI 的并行搜索技术，每个粒子在多维搜索空间中“飞翔”。在每个具体的时刻，每个粒子都有一个位置和速度。每个粒子的位置矢量代表了问题的一个可能解。

对于上述过程我们可以转化为一个数学问题：寻找函数 $y=1-\cos 3x \cdot e^{-x}$ 在[0，4]区间上的最大值，该函数的曲线如图 2.1 所示。

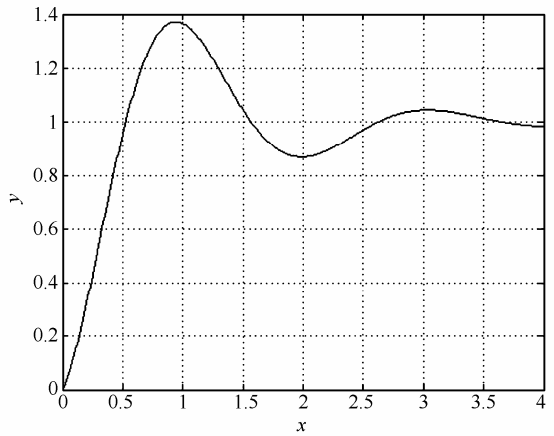

图 2.1　函数 $y=1-\cos 3x\ \mathrm{ge}^{-x}$ 在[0，4]区间上的曲线

当 x=0.9350～0.9450 时，达到最大值 y=1.3706。为了得到该函数的最大值，我们在[0，4]之间随机的洒一些点，为了演示，我们放置两个点，并且计算这两个点的函数值，同时给这两个点在[0，4]之间设置一个速度。下面这些点就会按照一定的公式更改自己的位置，到达新位置后，再计算这两个点的值，然后再按照一定的公式更新自己的位置。直到最后在 y=1.3706 这个点停止自己的更新。这个过程与基本 PSO 算法作为对照如下。

（1）这两个点就是基本 PSO 算法中的粒子。

（2）该函数的最大值就是鸟群中的食物。

（3）计算两个点的函数值就是基本 PSO 算法中的适应度值，计算用的函数就是基本 PSO 算法中的适应度函数。

（4）更新自己位置的公式就是基本 PSO 算法中的位置速度更新公式。

下面通过图形演示一下这个算法运行一次的大概过程，如图 2.2～图 2.7 所示。

以上用于说明基本 PSO 算法的例子实际上是一个函数优化的例子。由迭代过程图形的变化可以看出，两个点（粒子）在搜索空间［0，1］不断地调整自己的位置，使自己的函数值不断地向最大值靠近，最后，两个粒子都聚焦在最大值附近，寻找到了函数的近似全局最优解，说明了基本 PSO 算法的有效性。

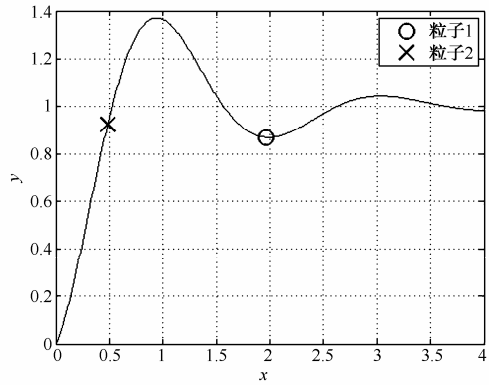

图 2.2　初始化粒子

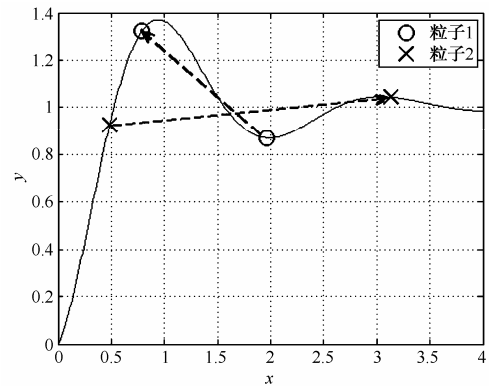

图 2.3　粒子位置第 1 次更新

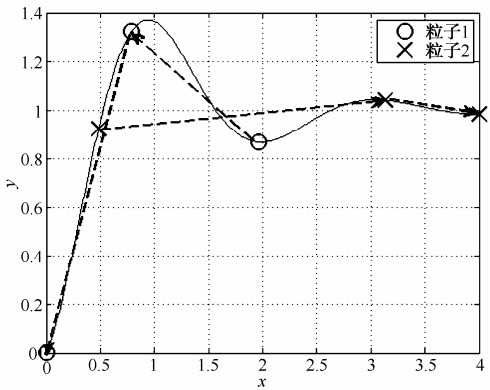

图 2.4　粒子位置第 2 次更新

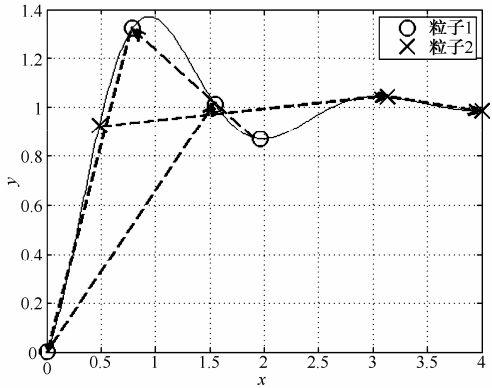

图 2.5　粒子位置第 3 次更新

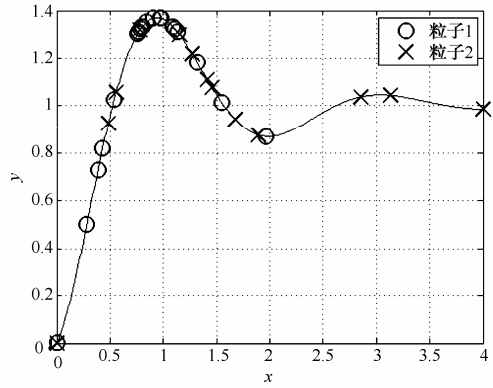

图 2.6　粒子位置第 20 次更新

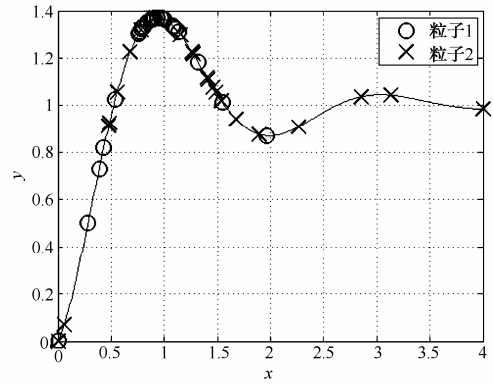

图 2.7　粒子位置第 30 次更新

Kennedy 和 Eberhart 最早也是通过函数优化来说明基本 PSO 算法的基本概念的[1]。n 维空间中的函数可用下式来描述：

$$f(x_1, x_2, x_3, \mathrm{L}\ , x_n) = f(\boldsymbol{X}) \tag{2.1}$$

其中，$\boldsymbol{X}$ 代表了函数所有可能的解向量。函数优化的目的就是在可能的解空间中寻找最优向量，使得其能够取得最大值和最小值，这个值我们可以用 f^* 来表示。对一些简单的函数，通过观察或简单传统的计算方法就可以确定它们的最优解（最大值或最小值）。但是对于许多函数来说，寻找它们最优解的工作是非常不容易的，比如对下面的函数：

$$f(x_1, x_2) = x_1 \sin(4\pi x_2) - x_2 \sin(4\pi x_1 + \pi) + 1 \tag{2.2}$$

这是一个多峰谷的函数，有一个非常粗糙的适应度值表面，如图 2.8 所示。利用传统的算法想要快速寻找这类函数的全局最优解显然是非常困难的，也是不切合实际的。因此，需要一种并行的搜索技术来实现复杂问题的优化问题。基本 PSO 算法就是基于群体智能（SI）的并行搜索技术，每个个体从不同的初始位置开始在搜索空间寻找函数的最优解，个体之间可以共享“信息”，从而在解的可行空间实现最优解的寻找。

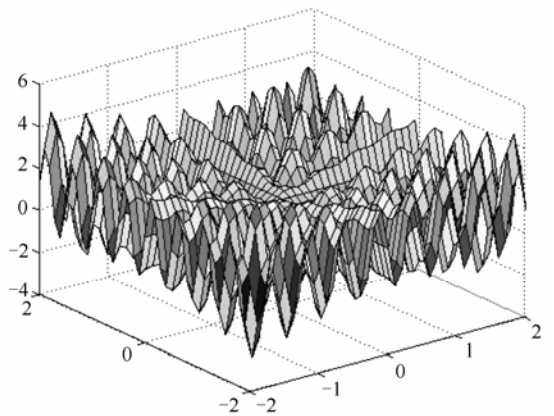

图 2.8　示例函数 3D 图形

在上面的例子中，没有给大家介绍我们取的这些随机点（粒子）是如何运动的，只是说按照一定的公式更新。这个公式就是基本 PSO 算法中的位置速度更新公式。下面就介绍这个公式，这个公式也是基本 PSO 算法的基础。

基本 PSO 算法是基于 SI 的并行优化搜索技术，种群（Swarm）由若干个粒子构成，粒子的个数称作种群大小或规模（Swarm Size）。每个粒子都有一个位置矢量和速度矢量，其维数代表了问题解空间的维数，分别记为：$\boldsymbol{X}_i$，$\boldsymbol{V}_i$。在前面求取函数 $y=1-\cos 3x\mathrm{ge}^{-x}$ 在区间[0，4]上的最大值时，在[0，4]之间放置了两个随机的点（粒子），在这个问题里，种群的大小为 2，粒子的维数为 1。更一般的是假设问题的搜索空间是一个 d 维空间，则粒子的位置矢量和速度矢量可以表示为：

$$\begin{aligned}\boldsymbol{X}_i&=[x_{i1},x_{i2},\mathrm{K},x_{id}]\\ \boldsymbol{V}_i&=[v_{i1},v_{i2},\mathrm{K},v_{id}]\end{aligned}\tag{2.3}$$

基于 PSO 算法初始化为一群随机粒子（随机解），每个粒子在搜索空间中“飞翔”，通过迭代找到最优解。在迭代过程中，粒子通过跟踪两个“极值”不断调整自己的位置，进行更新。第一个就是粒子本身所找到的最优解，这个解被称为“个体极值”或个体最优解，第 i 个粒子的个体最优解记为 $\boldsymbol{P}_\mathrm{i}=[p_{\mathrm{i}1},p_{\mathrm{i}2},\cdots,p_{\mathrm{id}}]$；另一个极值是整个种群目前所找到的最优解，称为全局极值或全局最优解，记为 $\boldsymbol{P}_g=[p_{g1},p_{g2},\cdots,p_{gd}]$。另外也可以不用整个种群而只是用其中一部分作为粒子的邻居，那么所有邻居中的极值就是局部极值。

例如在求函数前面求取函数 $y=1-\cos 3x\ \mathrm{g}\,\mathrm{e}^{-x}$ 在区间[0，4]上的最大值时，第 5 次迭代的图形如图 2.9 所示，很好地说明了个体极值和全局极值的概念。

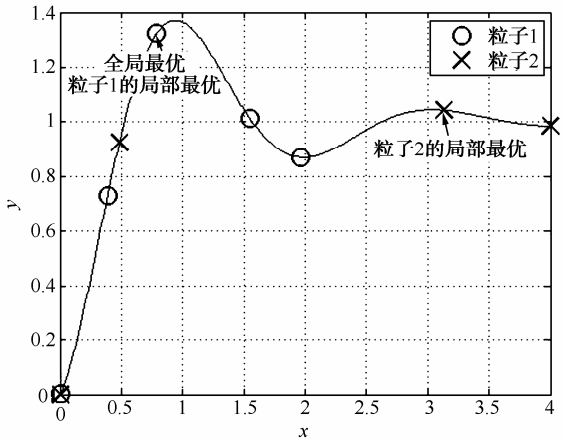

图 2.9　第 5 次迭代的图形

在找到这两个最优值时，粒子根据公式（2.4）来更新自己的速度和新位置：

$$\begin{aligned} v_{ij}(t+1) &= wv_{ij}(t) \\ &\quad + c_1 r_1[(p_{ij})_j(t) - x_{ij}(t)] \\ &\quad + c_2 r_2[p_{gj} - x_{ij}(t)] \\ x_{ij}(t+1) &= x_{ij}(t) + v_{ij}(t+1) \end{aligned} \tag{2.4}$$

$1 \leqslant i \leqslant n$，$1 \leqslant j \leqslant d$

其中，c_1、c_2 为正的常数，称为加速因子；r_1、r_2 为[0，1]之间的随机数；w 称为惯性因子。第 $j(1 \leqslant j \leqslant d)$维的位置变化范围和速度变化范围分别为 $[-x_{j,\max}, x_{j,\max}]$ 和 $[-v_{j,\max}, v_{j,\max}]$（变化范围可通过平移处理使之对称），迭代中若某一维的 x_{ij} 或 v_{ij} 超过边界则取边界值。粒子群初始位置和速度随机产生，然后按公式（2.4）进行迭代，直至满足停止条件。如图 2.10 所示为粒子速度更新公式（2.4）在 3D 空间的图解，所用的函数为：

$$f(x, y) = x^2 + y^2$$

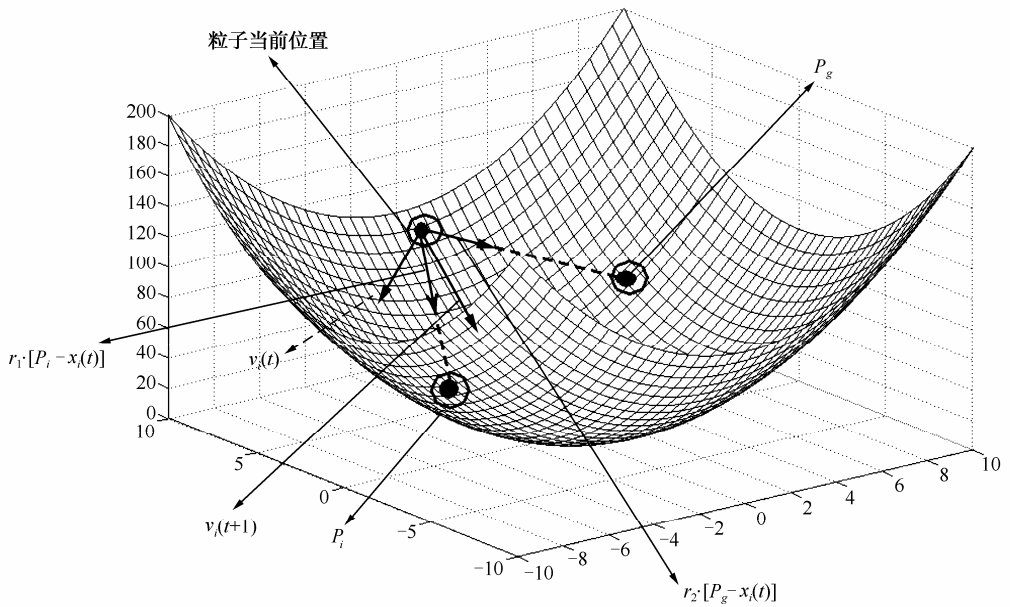

图 2.10　速度更新公式在 3D 空间的图解

从上述更新公式可以看出，粒子速度更新由三部分完成。

第一部分反映了粒子当前速度下时刻（或下一迭代步）的影响，联系粒子当前的状

态，是粒子惯性的表现，起到了平衡全局和局部搜索的能力。因此，称 w 为惯性因子。

第二部分反映了粒子对“自身经验”的认识，即粒子本身记忆的影响，这一部分使粒子具有全局搜索能力，避免陷入局部极小。因此，Venter 和 Socbeiski 也定义 c_1 为“自我认知（Self-Confidence）”。

第三部分反映了整个种群的“社会经验”，这部分综合考虑了以前迭代过程中整个种群所获得的“经验”，有利于粒子全局搜索能力的提升，因此 Venter 和 Socbeiski 也定义 c_2 为“群体认知（Swarm-Confidence）”。

目前，常用的基本 PSO 算法将全体粒子群(Global)分成若干个有部分粒子重叠的邻居子群（Local），这里的“邻居”一般不是指粒子位置之间的欧几里得空间相近，而是指粒子序号相邻，且这种邻居关系在整个迭代过程中通常一直保持不变。于是每个粒子根据 $\boldsymbol{P}_i$ 和邻居子群内历史最优 $\boldsymbol{P}_l=[p_{l1},p_{l2},\mathrm{L}\ \ p_{ld}]$ 调整自己的位置，即公式（2.4）中 p_{gj} 换为 p_{li}。基本 PSO 算法的流程如下所示。

Begin
（1）随机初始化种群中各粒子的速度和位置;
（2）根据适应度函数计算各粒子的适应度值，存储各粒子的位置和适应度值于个体极值中，将所有个体极值中适应度值最优的个体的位置和适应度值存储于全局极值中;
While（终止条件不满足）do
（1）按公式（2.4）更新各粒子的速度和位置;
（2）根据适应度函数重新计算各粒子的适应度值;
（3）更新粒子的人个体极值和全局极值：如果粒子当前的适应度值优于个体极值，则用粒子当前的位置和适应度值更新为当前值；如果种群中适应度值最优的粒子的适应度值优于全局极值，则用适应度值最优粒子的位置和适应度值更新全局极值。
EndWhile
记录全局极值等数据。
End

如图 2.11 至图 2.14 所示，通过 Rastrigin 函数在三维空间寻找最小值的过程来说明粒子群算法的动态更新过程（分别展示了迭代 50 步、100 步、150 步、200 步的运动过程）。

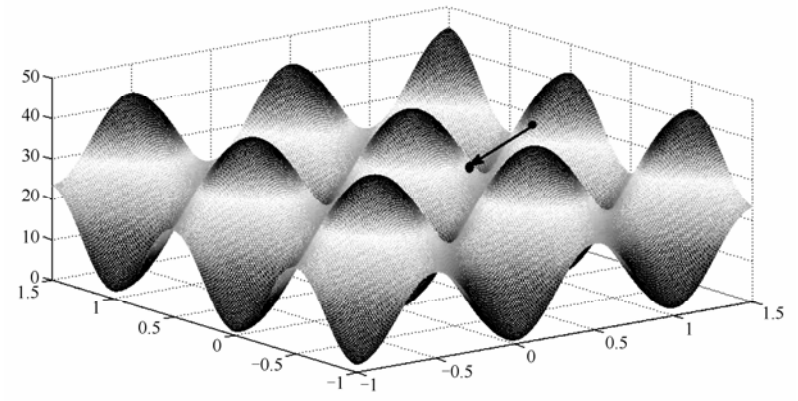

图 2.11　迭代 50 的运动过程

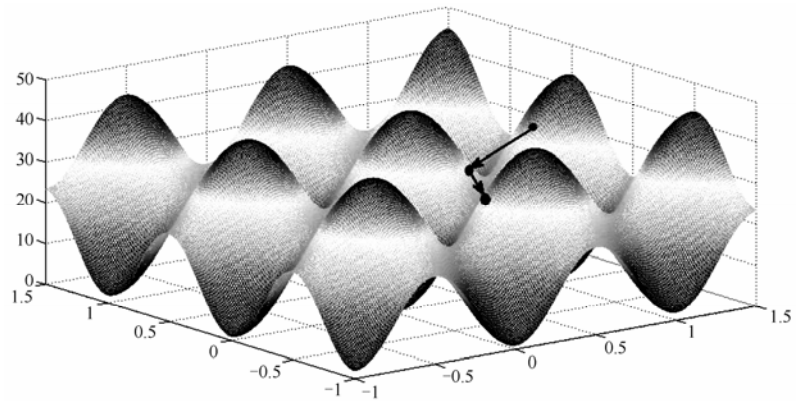

图 2.12　迭代 100 的运动过程

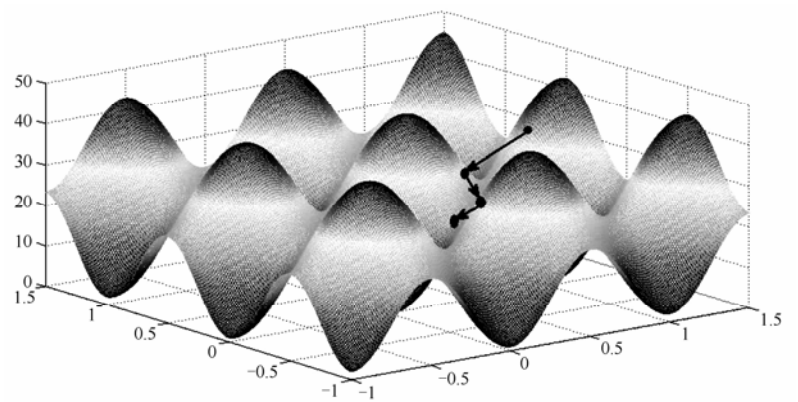

图 2.13　迭代 150 的运动过程

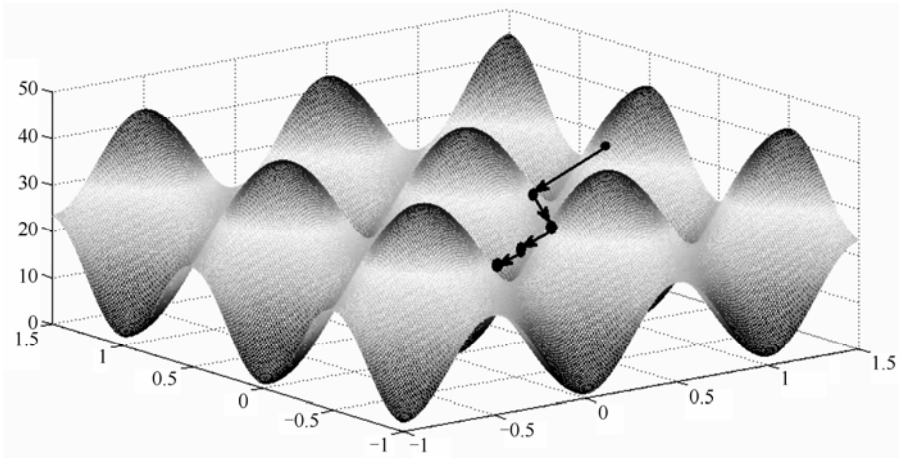

图 2.14　迭代 200 的运动过程

由以上粒子的运动过程可以看出，在迭代过程中，粒子根据公式（2.4）和函数值（适应度值）不断调整自己在空间的位置，向全局最优解（函数最小值）靠近，这是一个动态更新过程。

2.3　基本PSO算法设计原则及步骤

2.3.1　基本PSO算法设计原则

1．适用性原则

针对需要解决的优化问题类，根据其限制与假设以及目标要求，抽取其性质特点，寻找或设计一个适合该问题类的高性能 PSO 算法，不同的问题类的具体处理方式也不同。

2．可靠性原则

算法的可靠性是指算法经过有限步的运算之后，能对所处理的问题获得满足要求的解的能力。由于 PSO 算法是一种随机优化算法，因此在求解不同问题时，其结果具有一定的随机性与不确定性。因此设计算法实验时，一定要进行多次测试验证，以确定算法的可靠性。

3．收敛性原则

PSO 算法的收敛性是指算法能否以一定的收敛速度和收敛精度以概率 1 收敛到问题的全局最优解。算法的收敛性能可通过比较在有限时间内算法所求得解的精度是否满足问题目标解的精度要求来评价。

4．稳定性原则

算法的稳定性也指算法的健壮性，是指算法对其控制参数及问题数据的敏感程度。性能稳定的算法应同时具有以下两种特性：一是对一组固定的控制参数，算法适用于较广泛问题的求解；二是对给定的问题数据，算法的求解结果应不会随控制参数的微小扰动而波动。

5．生物类比原则

PSO 算法思想直接来源于对生物群体社会行为的模拟，因此可通过研究其他仿生算法的原理与机制，优化 PSO 算法，提高算法性能。

2.3.2　基本PSO算法步骤

1．确定问题的表示方案

同其他进化算法一样，PSO 算法在求解问题时，应首先将问题的解从解空间映射到具有某种结构的表示空间。由于 PSO 算法的应用主要集中在连续函数优化领域中，算法大多采用实数向量的编码方案。但是用 PSO 算法求解具有离散特征的问题时，应根据问题的特征选择适当的编码方法，提高算法的性能。

2．确定优化问题的目标函数

根据所求问题的具体特征，选取适当的目标函数来计算适应度，借助于适应度值来评价解的质量。适应度是唯一能够反映并引导算法优化过程不断进行的参量。

3．选取算法参数

对 PSO 算法性能有显著影响的参数有：群体规模、惯性因子、学习因子、最大速度、最大迭代次数 N。适当的选取算法控制参数值有助于改善算法的性能。

4．设计粒子的飞行模型

在 PSO 算法中，粒子的速度大小以及变化范围直接影响着粒子的寻优能力。粒子在飞行（寻优）过程中，借助于个体的局部最优与群体的全局最优动态地调整自己的飞行速度与方向。速度太快，则粒子有可能越过全局极极值点；太慢，则粒子有可能陷入到局部极值区域内。

5．确定算法的终止准则

PSO 算法中最常用的终止准则有：设定最大允许迭代次数；设定最大允许迭代次数和最优解精度误差；设定最优解 L 步误差精度范围，即在迭代过程中最优解的适应度在连续 L 步后变化误差范围小于给定的误差精度。

6．编写程序，上机调试运行

经过以上步骤后，即可确定思路，编写程序并调试运行。

2.4　基本PSO算法与其他算法的比较

基本 PSO 算法与其他进化算法相比，有其共性又有其不同。相同之处可以归结为以下几点。

（1）同其他进化算法相比，它们最大的相同点是通过群体的概念来寻找最优解。在算法的进化过程中通过大量的个体行为聚合，并形成群体行为来寻找目标值，遗传算法 GA 和蚁群优化（ACO）算法均是如此。

（2）基本 PSO 算法同其他进化算法的一样，都对函数本身的解析性质无特殊要求，仅通过适应度的概念来表示粒子是否满足要求的条件。该性质是整个现代算法的一个共有特性，由于本身具有较强的适应性和实用性，而常常被作为算法研究的基础。纵观目前所有的进化算法和群体智能（SI）算法，均是如此。

（3）同其他进化算法一样，算法中个体的进化过程都是通过迭代更新完成，如 GA、ACO 算法等，通过个体的不断进化最终达到整个群体的进化。

同样每种算法都有各自独有的一些特点，基本 PSO 算法与其他进化算法的主要不同表现在两点。

（1）算法思想不同。每种算法都有其思想，GA 通过对解编码将每个值转化为二进制序列，而每个二进制序列是不相同的，但是每个二进制序列的更新与其他二进制序列没有直接影响；ACO 算法中的蚂蚁都是按照“信息素”的多少寻找

路径，并在走过的路径上留下自身的信息素；基本 PSO 算法是从鸟群的群体行为出发的模型，因此它的思想是根据鸟群的独有特性，群中每个鸟都会受到其他鸟的直接或间接影。

（2）种群中个体的更新方式不同。基本 PSO 算法直接来源于鸟群捕食行为的研究。在捕食过程中，每个鸟都会受到其他鸟的影响，因此算法本身考虑到了两部分信息：自身信息和群体信息。每个粒子都是通过这两部分信息来确定它的运动趋势，这主要体现在基本 PSO 算法的更新模型上，这是基本 PSO 算法与其他算法的一个显著的不同之处，也是基本 PSO 算法能够表现出比其他进化算法具有更多优良特性的重要原因。

2.5　基本PSO算法参数的选择

PSO 算法速度更新公式中的主要参数包括：惯性因子 w、加速因子 c_1 和 c_2、种群大小 S 和粒子的最大速度 $v_{\max}$。优化问题是多种多样的，同样的参数设置适应于所有的优化问题这是不可能。因此，这些参数的设置决定了粒子群优化算法在具体应用中的性能。目前国内外很多研究人员为了提高算法的优化性能，对粒子群算法的参数的选择和设置规则进行了大量的研究和实验。

1．惯性因子

在最初的基本 PSO 算法中，没有惯性因子 w，$v_{\max}$ 对算法性能的影响非常大，与优化问题相关性非常强，且不存在经验值，不合适的 $v_{\max}$ 将导致系统发散。Shi 和 Eberhart 于 1998 年提出了惯性因子 w 的概念[7]，通过惯性因子 w 可很好地控制粒子的搜索范围，大大削弱了 $v_{\max}$ 的重要性。w 值较大，全局寻优能力强，局部寻优能力弱；w 值较小，则利于局部搜索。如果 $w=0$，则粒子速度只取决于它当前位置的个体极值和全局极值，速度本身没有记忆。假设一个粒子位于全局最优位置，它将保持静止。而其他粒子则飞向其本身的个体极值和全局极值的加权中心。这种条件下，粒子群将收缩到当前全局最优位置，更像一个局部算法。如果 $w\neq 0$，则粒子有扩展搜索空间的趋势，从而针对不同搜索问题，可调整算法的全局和局部搜索能力。惯性因子 w 的引入使 PSO 算法的性能得到了很大提高，也使 PSO 算法得以比较成功地应用于很多实际问题。

Eberhart 和 Shi 在几篇文章中探讨了惯性因子的设置问题[8,9,10]。初始时，Shi 将 w 取为常数，发现在最大速度不是太小的情况下（$v_{\max}\geqslant 3$），惯性因子 $w=0.8$ 是一个比较好的选择。虽然这个结论仅仅是对 Schaffer 第 6 函数进行测试的结果，但实践

证实，这个结论在许多问题上确实是成立的。后来的实验发现，动态 w 值能够获得比固定值更好的寻优结果，w 可以在 PSO 算法搜索过程中线性变化，也可根据 PSO 算法性能的某个测度函数动态改变。目前，采用较多的是 Shi 建议的线性递减权值(Linearly Decreasing Weight，LDW)策略，即

$$w(k) = w_{\min} + \frac{w_{\max} - w_{\min}}{N} \times (N - k) \tag{2.5}$$

其中，N 为最大进化代数，$w_{\min}$ 和 $w_{\max}$ 分别为最小、最大惯性因子。典型取值为 $w_{\min} = 0.4$，$w_{\max} = 0.9$。Eberhart 和 Shi 还设计了一种自适应模糊 PSO 算法，用一个模糊控制器去控制惯性因子随时间变化。

2．收缩因子

2002 年，Clecr 和 Kennedy 提出了自适应 PSO 算法[11]，引入了收缩因子（Constriction Factor）χ来保证 PSO 算法收敛，这也是另一个版本的标准算法。在这个算法中，忽略了惯性因子 w 和最大速度 $v_{\max}$，第 i 个粒子的 d 维速度更新公式变为：

$$\begin{aligned} v_{ij}(t+1) &= \chi[v_{ij}(t) + c_1 r_1[p_{ij}(t) - x_{ij}(t)] + c_2 r_2[p_{gj} - x_{ij}(t)]] \\ x_{ij}(t+1) &= x_{ij}(t) + v_{ij}(t+1) \end{aligned} \tag{2.6}$$

$1 \leqslant i \leqslant n$，$1 \leqslant j \leqslant d$

其中，收缩因子

$$\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}，\quad \varphi = c_1 + c_2，\quad \varphi > 4 \tag{2.7}$$

在使用收缩因子方法时，通常取 $\varphi = 4.1$，从而使收缩因子χ=0。729。Clecr 在推导出收缩因子法时，不再需要最大速度限制 $v_{\max}$。但是，后来研究发现设定最大速度限制可以提高算法的性能。从数学上分析，惯性因子 w 和收缩因子χ这两个参数是等价的。

3．最大速度

最大速度 $v_{\max}$ 决定着粒子在迭代过程中在位置坐标所能允许的最大变化。为了防止系统迭代中出现发散，需要对粒子的最大速度进行限制，因此粒子最大速度 $v_{\max}$ 是一个非常重要的参数。$v_{\max}$ 太大，可能会使得粒子飞过最优解；而太小的 $v_{\max}$ 则易导致粒子搜索速度太慢，或是被局部最优解所吸引，无法找到最优解。在早期的 PSO

算法实验中，加速因子 c_1 和 c_2 几乎都采用 c_1= c_2= 2，而惯性因子 w 也是固定为 1，所以 $v_{\max}$ 成了实验中唯一需要调整的参数。实验发现，采用 $v_{\max}=x_{\max}-x_{\min}$ 为粒子最大速度通常可取得较好的效果，因此目前的 PSO 算法也一般采用这个粒子最大速度约束条件。例如，如果粒子的位置矢量为 $\boldsymbol{X}=[x_1,x_2,x_3]$，并且$-10\leqslant x_i\leqslant 10$（$i=1,2,3$），有 $v_{\max}=20$。最初，引入粒子的最大速度的上限是为了防止系统在迭代过程中发散，随着惯性因子和收缩因子被用于速度更新公式中，最大速度 $v_{\max}$ 在某种程度上不再显得重要，至少没有它，算法的收敛性还是能够得到保证的。因此，一些研究者为了简化程序设计不再对速度进行限制，尽管这样是允许的，但是最大速度 $v_{\max}$ 的限制在许多时候还是能够改善算法的优化性能。

4．群体规模

群体规模 M 大小的选择没有明确的公式依据，一般采用经验法取 20～60[8,12]，对较难或特定类别的问题可以取到 100～200。Vanden Bergh 和 Engelbrecht 的实验结果表明[13]，增大 M 对改善算法的收敛精度的效果并不明显，而算法的计算复杂度却随着 M 的增大而快速增加，反而达不到寻优效果；Eberhart 和 Shi 也证明种群大小几乎不会影响 PSO 算法的性能[14]。

5．加速因子

加速因子 c_1 和 c_2 代表将每个粒子推向个体极值和全局极值位置的统计加速项的权值。较低的值允许粒子在被拉回之前可以在目标区域外徘徊，较高的值导致粒子突然地冲向或越过目标区域。如果令 c_1=c_2=0，粒子将一直以当前速度飞行，直至边界，很难找到最优解。

当 c_1=0 时，则粒子没有了认知能力，变为“只有社会”（Social-Only）的模型，

$$v_{ij}(t+1)=wv_{ij}(t)+c_2r_2[p_{gj}-x_{ij}(t)] \tag{2.8}$$

此时称为全局 PSO 算法。粒子有扩展搜索空间的能力，具有较快的收敛速度，但由于缺少局部搜索，对于复杂问题比基本 PSO 算法更易陷入局部最优。

当 c_2=0 时，则粒子之间没有社会信息，模型变为“只有认知”（Cognition-Only）模型，即

$$v_{ij}(t+1)=wv_{ij}(t)+c_1r_1[p_{ij}-x_{ij}(t)] \tag{2.9}$$

此时称为局部 PSO 算法。由于个体之间没有信息的交流，整个群体相当于多个粒子进行盲目的随机搜索，收敛速度慢，因而得到最优解的可能性小。

早期应用中，通常设 c_1=c_2=2。不过 Suganthan 的实验表明：c_1 和 c_2 为常数时可以得到较好的解，但不一定必须等于 2，可以在[0，4]中取值。Ratnaweera 等人最近研究了加速因子随时间改变时对 PSO 算法的影响[15]。他们通过公式（2.10）改变加速因子 c_1 和 c_2：

$$\begin{aligned} c_1 &= (c_{1f} - c_{1i})\frac{n_{\text{iter}}}{\max n_{iter}} + c_{1i} \\ c_2 &= (c_{2f} - c_{2i})\frac{n_{\text{iter}}}{\max n_{iter}} + c_{2i} \end{aligned} \tag{2.10}$$

其中，c_{1i}、c_{1f}、c_{2i} 和 c_{2f} 为常量，*iter* 是当前迭代次数，maxn_{iter}。

结果表明，这种方法在算法早期增强了全局搜索能力，有利于在搜索结束时收敛到全局最优。这种方法被命名为 PSO-TVAC（PSO with Time Varying Acceleration Coefficients）算法。

6．种群拓扑结构

常见的 PSO 算法有两个版本：全局版本的 PSO 算法和局部版本的 PSO 算法。在全局版本的 PSO 算法中，每一个粒子根据自己的全局最优解和整个种群的全局最优解在搜索空间调整。在局部版本的 PSO 算法中，每个粒子根据自己的全局最优和它邻域内粒子的全局最优解调整自己的位置。粒子的邻域从拓扑学上被定义为粒子周围离它最近的那个粒子。当然如果考虑全局版本的粒子群中相邻的两个粒子互为邻居的话，它也可以作为局部版本的粒子群使用。Kennedy[1]认为全局版本的 PSO 算法具有较快的收敛速度，但是容易陷入局部最优；局部版本的 PSO 算法可能更容易寻找到全局最优，但收敛速度较慢。因此，许多研究者都试图通过对种群拓扑结构的研究来改善 PSO 算法的性能。不过，Kennedy 认为有较少邻居的粒子群在解决复杂的优化问题时有更好的性能，而具有较多邻居的粒子群更适合处理简单的优化问题[12]。

种群拓扑结构是指整个种群所有粒子之间的联结方式(相互联结的粒子进行通信)。而邻域结构则是单个粒子与它通信粒子的联结方式。由此定义可知，粒子的行为主要由其局部邻域结构影响，该局部邻域可视为种群拓扑结构中的部分区域。种群邻域结构的限定可阻止信息在整个种群中的流动，从而保持种群多样性，它可控制算法的探测和开发能力。种群拓扑结构对 PSO 算法性能的影响有两个层面：其一，可选取不同粒子的局部邻域结构；其二，可定义不同的局部邻域之间的通信方式。

由于基本 PSO 算法存在易于陷入局部最优解的缺陷，Kennedy 从社会学的角度对此进行了分析和研究，他认为导致算法早熟的原因是粒子间的信息传播速度过于迅速。在社会学中有一个术语“Small Worlds”，是指社会中一个人能很容易地间接分享大量其他人的信息。Kennedy 因此提出了局部版的 PSO 算法[12]。任一个粒子 i 与(i−l)和(i+l)两个粒子形成邻居，粒子 i 的“社会经验”来自于它的邻居中，这样的粒子群就有着一种环型的邻居拓扑结构。这种拓扑结构一定程度上摆脱了基本 PSO 算法易于陷入局部最优解的缺点。

比较全局和局部版本两种算法，可注意到，它们收敛速度和跳出局部最优解的能力有所差异。由于全局拓扑结构中所有粒子都信息共享，粒子向当前最优解收敛的趋势非常显著，因而全局模型通常收敛到最优解的速度较局部结构快，但更易陷入局部最优解，表现为整个种群一致收敛到当前第一个最优解。局部拓扑结构模型则允许粒子与其邻居比较当前搜索到的最优位置，从而相互之间施加影响，即便其值比种群最优解要差，该影响可以使较差个体进化为较好的个体。

随后 Kennedy 又提出了轮型拓扑结构，和一些不同的个体极值拓扑结构，如图 2.15 所示，Mendes 通过实验对 10 种邻居结构作了比较分析[16]。

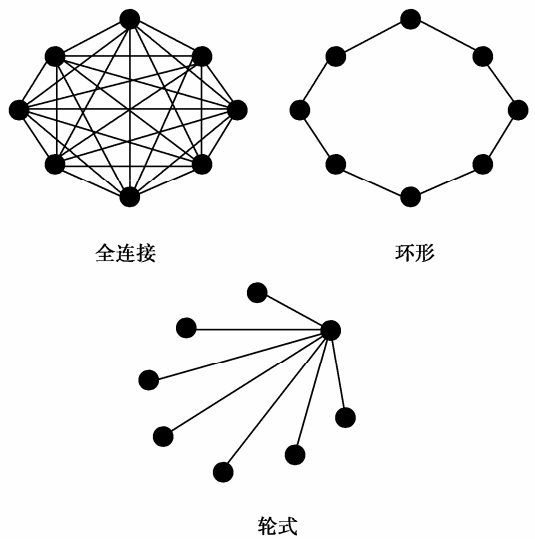

图 2.15　常见的几种邻域拓扑结构

Mendes 和 Kennedy 近期提出了 Fully Informed PSO 算法[17, 18]，认为基本 PSO 算法过于强调邻居中最佳个体的作用和意义，主张将邻居内所有粒子的信息作为“社

会经验”，以邻居粒子的适应度值或距离作为权值，用邻居内所有粒子的加权平均替换个体极值。不依赖最佳个体，而是相信集体经验，这种粒子邻居间的行为更符合和贴近人类的从众社会行为。

参 考 文 献

[1] Kennedy J, Eberhart R.C. Particle Swarm Optimization.In:IEEE International Conference on Neural Networks,IV.Piscataway,NJ:IEEE Service Center,1995.1942～1948.

[2] 米歇尔·沃尔德罗[美]．复杂——诞生于秩序与混沌边缘的科学．北京：三联书店，1997.

[3] Reynolds C.W., Flocks Herds and Schools. A Distributed Behavioral Model.Computer Graphics, 1987, 21(4):25～34.

[4] Langton C.G. Artificial Life: an overview. MIT Press, 1995.

[5] 曾建潮，介倩，崔志华．微粒群算法[M]．北京：科学出版社，2004.

[6] 宋胜利．合粒子群协同优化算法及其应用研究．华中科技大学博士论文，2009.

[7] Shi Y，Ebethart R.C. A modified Optimizer. Proceedings of the IEEE Congresson Evolutionary Computation. Piseataway，NJ.1998，69～73.

[8] Shi Y，Eberhart R.C. A Modified Particle Swarm Optimiser. IEEE International Conference on Evolutionary Computation (1998), Anchorage, Alaska, May，4～9.

[9] Shi Y，Eberhart R.C. Fuzzy Adaptive Particle Swarm Optizimation. In Proceedings of the Congress on Evolutionary Computation 2001. Seoul, Korea, IEEE Service Center, IEEE(2001), pp. 101～106.

[10] Shi Y，Eberhart R.C. Comparing Inertia Weights and Constriction Factro in Particle Swarm Optizimation. In Proceedings of the IEEE International Congress on Evolutionary Computation, vol. 1, pp. 84～88.

[11] Clecr M，Kennedy J. The Particle Swarm – Explosion, Stability and Covergence in a Multidimensional Complex Space. IEEE Transactions on Evolutionary Computation,6(1), 58～73.

[12] Kennedy J. Small Worlds and Mega-Minds: Effects of Neighborhood Toplogy on Particle Swarm performance. In Proceedings of the 1999 Congress on Evolutionary Computation , vol. 3., IEEE Press, New York, pp. 1931～1938.

[13] Vanden Bergh F，Engelbrecht PA. Effects Swarm Size on Cooperative Particle

Swarm Optimizers. In Proceedings of GECCO-2001, San Francisco, CA, pp. 892～899.

[14] Kennedy J. Small Worlds and Mega-Minds: Effects of Neighborhood Toplogy on Particle Swarm performance. In Proceedings of the 1999 Congress on Evolutionary Computation , vol. 3., IEEE Press, New York, pp. 1931～1938.

[15] Ratnaweera A, Halgamuge SK, Watsom H.C. Self-Organizing Hierarchical Particle Swarm Optimizer with Time-Varying Acceleration Coefficients. IEEE Transactions on Evolutionary Computation, 8(3), 240～255.

[16] Kennedy J,R Mendes.Topological Structure and Particle Swarm Performance.In The 2002 Congress on Evolutionary Computation.Piscataway,NJ:IEEE Service Center,2002.1671～1676.

[17] Mendes R.Population Topologies and Their Influence in Particle Swarm Performance Department of Information,University of Minho,Braga,Portugal.2004.

第 3 章　改进的PSO算法

粒子群优化（PSO）算法作为一种基于群体智能（SI）的优化技术，原理简单，容易实现，具有较强的通用性和全局寻优的特点，是求解非线性和多峰特性目标函数全局最优问题的一种有效方法。但是基本 PSO 算法也存在局部搜索能力较差，搜索精度不高，不能够保证搜索到全局最优解，容易陷入局部最优解，对参数具有一定的依赖性等诸多的不足。因此，许多学者和研究人员在基本 PSO 算法的基础上从参数的设置、种群拓扑结构、与其他算法的混合等方面，提出了很多改进的 PSO 算法。这里介绍一些典型的改进了的 PSO 算法，并对这些算法的性能的应用领域进行了探讨。

3.1　离散PSO算法

PSO 算法最初被用于连续问题求解，近年来其在离散优化问题中的应用日益引起人们的注意，出现了一些离散 PSO（Discrete PSO，DPSO）算法。

为了用 PSO 算法求解离散组合优化问题，人们通常采用两种方法。

（1）以基本的连续 PSO 算法为基础，针对特定问题，将离散问题空间映射到连续粒子运动空间，并适当修改 PSO 算法来求解，在计算上仍保留经典 PSO 算法在速度-位置更新中的连续运算规则。

（2）针对离散优化问题，以 PSO 算法信息更新的本质机理为基础，在基本 PSO 算法的基本思想、算法框架下，重新定义特有的粒子群离散表示方式与操作算子来求解。在计算上以离散空间特有的对矢量中的位操作取代传统向量计算。从信息流动机制上看，仍保留了 PSO 算法特有的信息交换和流动机制。

这两种方法的区别在于：前者将实际离散问题映射到粒子连续运动空间后，在连续空间中计算并求解；后者则是将 PSO 算法映射到离散空间，在离散空间中计算和求解。根据其特性，将前者称为基于连续空间的 DPSO 算法，后者称为基于离散空间的 DPSO 算法。

3.1.1　二进制PSO算法

由于基本 PSO 算法主要针对连续函数进行搜索运算，但许多实际工程问题都描述为离散的组合优化问题，为此 Kennedy 和 Eberhart 于 1997 年在基本 PSO 算法的

基础上提出了二进制 PSO（Binary Particle Swarm Optimization，BPSO）[1]算法，这是基于连续空间的 DPSO 方法。

在 BPSO 算法中，粒子的搜索空间相应变为 n 维的二进制空间，记为 $\boldsymbol{B}^n$。同时，仍然需要定义适应度函数 f 对粒子进行更新，适应度函数根据粒子的位置矢量计算得到实数形式的适应度值，即 $f:\boldsymbol{B}^n\rightarrow\boldsymbol{R}^n$。为了将 PSO 算法离散化，BPSO 算法中每个粒子的位置矢量属于二进制空间 $\boldsymbol{B}^n$（由 0 和 1 组成的二进制字符串），但是它的速度矢量仍然属于实数空间 $\boldsymbol{R}^n$，即有 $V\in[-V_{\max},V_{\max}]\subset\boldsymbol{R}$。这样，BPSO 算法的公式可以通过保留基本 PSO 算法中的速度公式而只对位置更新公式进行修改得到。每个粒子的速度和位置更新公式为

$$v_{ij}(t+1)=wv_{ij}(t)+c_1r_1[p_{ij}(t)-x_{ij}(t)]+c_2r_2(p_{gj}-x_{ij}(t)) \tag{3.1}$$

$$x_{ij}(t+1)=\begin{cases}0, & \text{其他}\\ 1, & \rho<s[v_{ij}(t+1)]\end{cases} \tag{3.2}$$

其中，ρ 是[0,1]之间的随机数，算法中其他参数都和基本 PSO 算法内的参数相同。算法由当前的速度变量决定粒子将被判定为 1 或 0 的概率：

$$\boldsymbol{P}[x_{ij}(t+1)=1]=s[v_{ij}(t+1)] \tag{3.3}$$

即由粒子速度决定一个范围在[0,1]之间的概率选择参数 s：若 s 接近于 1，则粒子将更可能被选择为 1；若 s 接近于 0，则粒子更可能被选择为 0。Kennedy 等提出使用 Sigmoid 函数求参数 s。Sigmoid 函数是神经网络中常用的一种模糊函数，其表达式为：

$$s=S(v_{ij}(t))=\frac{1}{1+\mathrm{e}^{-v_{ij}(t)}} \tag{3.4}$$

当取得 $v_{\max}=6$ 时，阈值 s 的取值范围为[0.0025,0.9975]。

实验显示，对于大多数测试函数，BPSO 都比遗传算法（GA）速度快，尤其在问题的维数增加时。

BPSO 算法并不直接优化二进制变量本身，而是通过优化连续变化的二进制变量为 1 的概率，达到间接优化离散变量的目的。由于许多组合优化问题中存在序结构表达和约束条件处理等问题，BPSO 不能完全适用。一些研究者提出了对粒子位置的近似取整策略，即对迭代产生的连续解进行取整运算并以此评价解的质量，常采用向上取整和舍尾取整。这种算法的核心就是对粒子位置进行取整计算。取整策略是连续 PSO 算法的一种直接离散化方式，算法的其余部分完全继承连续 PSO 算法，在使用时不用改动。

用基于连续空间的 DPSO 求解离散问题时，算法生成的连续解与整数规划问题的目标函数评价值之间存在多对一的映射，因此该目标函数不能完全反映 PSO 算法中连续解的质量。此外整数规划问题的连续化导致大量冗余解空间与冗余搜索，从而影响算法的收敛速度。但是由于连续空间里的矢量计算十分简单，消耗时间短，因此，基于连续空间的 DPSO 算法仍能保持较快的运算速度。建立问题解到粒子位置空间的映射，是应用这一类 DPSO 解决问题的关键，除 0-1 松弛化、近似取整外的其他策略也是目前研究的方向之一。

3.1.2　基于离散空间的DPSO算法

目前关于基于离散空间的 DPSO 算法的研究还较少。研究者们往往根据具体问题，构建相应的粒子表达方式，并通过重新定义粒子更新公式中的加减法和乘法运算规则来求解。如 Clerc[2]针对旅行商问题（Traveling Salesman Problem，TSP）提出的 TSP-DPSO 算法，Farza-neh[3]针对 0-1 规划问题提出的离散 BPSO 算法。

TSP-DPSO 算法中，用所有城市的一个排列来表示粒子的一个位置，所有的排列就构成了问题搜索空间。并引入交换子和交换序列概念：一个交换子 $S = S_{\text{wap}}(i, j)$，就是交换位置中第 i 个和第 j 个元素，一组特定顺序的交换子集合称为一个交换序列 $S_S = (S_1, S_2, \text{L}\ , S_m) = ((i_1, j_1), (i_2, j_2), \text{L}\ , (i_m, j_m))$。速度则定义为粒子为达到目标状态所需要对其当前位置状态执行的基本交换序，例如粒子位置 $\boldsymbol{P}_1$、$\boldsymbol{P}_2$ 分别为（1，2，3，4，5）和（3，1，5，2，4）时，对应速度 $\boldsymbol{V} = \boldsymbol{P}_2 - \boldsymbol{P}_1$ 为交换序列（（1，3），（2，3），（3，5），（4，5））。此时,粒子的位置和速度状态不再是同维量。基于这一概念，Clerc 重新定义了 PSO 算法中的“加减法”操作，并定义速度与随机数的数乘为依随机数对应概率值保留速度中的所有交换子，实现了 PSO 算法向离散空间的映射。其引入了基本交换序概念，即所有作用于同一解上会产生相同新解的等价交换序中，拥有最少交换子的交换序称为基本交换序列。每次更新速度后，即重新计算速度 $\boldsymbol{V}(t+1)$ 的等价基本交换序，作为粒子新的历史速度，以避免随算法的迭代造成速度记录列表的累积。重新定义后的粒子状态更新公式为：

$$\begin{aligned} \boldsymbol{V}_i(t+1) &= \boldsymbol{V}_i(t) \oplus c_1 \oplus [P_i - X_i(t)] \oplus c_2 \oplus [P_g - X_i(t)] \\ \boldsymbol{X}_i(t+1) &= \boldsymbol{V}_i(t+1) \oplus \boldsymbol{X}_i(t) \end{aligned} \qquad (3.5)$$

离散 BPSO 算法中，定义粒子的位置和速度为由 0 和 1 组成的同维度矢量，因子 c_1、c_2 为随机生成的与位置同维度的矢量，矢量间的加减法为对二进制位的异或操作，记为 $\oplus$，矢量间乘法为对二进制位的与操作，记为 $\otimes$，从而构造了一种离散空间的新型 BPSO 算法。其还借鉴了免疫机制以避免算法陷入局部最优，从其实验

结果看离散 BPSO 算法比基于连续空间的 BPSO 算法和遗传算法（GA）的效率都高。离散 BPSO 算法中粒子速度是与位置同维的二进制矢量，粒子的更新计算在离散空间中进行，这与基于连续空间的 BPSO 算法完全不同。

基于离散空间的离散 DPSO 使用了位操作，可能会增加单步计算代价，但不存在冗余搜索问题，且对离散问题表达自然，易于与其他演化算法结合，发展前景很好。但现有研究主要针对个别类型问题，缺少一个统一的、通用的标准模型。

3.1.3　改进的BPSO算法

BPSO 算法虽然收敛速度较快，但仍有可能出现早熟现象，从而无法搜索到全局最优解。为此，一些研究者基于基本 BPSO 算法提出了许多改进的离散 PSO 算法。

1．基于分布估计的离散PSO算法

周雅兰等[4]把分布估计算法思想引入到 PSO 算法中，提出一种基于分布估计的离散 PSO（Estimation of Distribution PSO，EDPSO）算法。在 EDPSO 算法中，第 t 次迭代时粒子 i 的位置 $x_{ij}(t)\in\{0,1\}$。首先统计个体历史极值的每一维出现 1 的个体数量，根据概率模型计算出一个实值概率向量，表示为 $\boldsymbol{P}=(p_1,p_2,\mathrm{L}\ ,p_d,\mathrm{L}\ \ p_d)$，$p_j$ 表示粒子的第 j 维取值为 1 的概率。然后，这个概率向量 $\boldsymbol{P}$ 引导粒子在[0，1]的解空间进行搜索。在产生下一代种群时，以大概率 β 从概率向量抽样产生新解，以小概率 $1-\beta$ 直接复制种群全局最优解作为新解。即个体粒子的部分维值由概率向量来决定，部分维值直接来源于群体最优个体。EDPSO 算法产生新个体粒子的机制可描述如下。

如果　$\mathrm{rand}()<\beta$

$$\begin{cases} x_{ij}(t+1)=1, & \mathrm{rand}()<p_j \\ x_{ij}(t+1)=0, & \text{其他} \end{cases} \tag{3.6}$$

其他

$$x_{ij}(t+1)=p_{gj}(t) \tag{3.7}$$

其中，rand()是[0,1]区间上的随机数，粒子的每一维 x_{ij} 值由参数 β 控制是依据概率向量 $\boldsymbol{P}$ 产生的，还是依据全局最优值 $\boldsymbol{P}_g$ 产生的。EDPSO 算法的具体流程和步骤如下所示。

```
Begin
  初始化种群，并保存所有粒子的个体历史;
```

用下式初始化概率向量 $\boldsymbol{P}$

$$p_j = \frac{\sum_{i=1}^{N} p_{ij}}{N}$$

While（终止准则不满足）do

根据公式（3.6）和（3.7）生成新个体粒子;

按下式进行变异操作（α为变异概率），生成新一代种群

$$x_{ij}(t) = 1 - x_{ij}(t)，\text{条件：}\quad \text{rand}() < \alpha$$

评价新种群中所有粒子的适应度;

比较适应度值更新当前的个体历史极值和全局最优值;

使用下式更新概率向量 $\boldsymbol{P}$

$$p_j = (1-\lambda)p_j + \lambda\, \mathrm{g}\frac{\sum_{i=1}^{N} p_{ij}}{N}$$

End While

记录全局最优值和其他数据。

End

为了验证提出算法的有效性，在二分图问题上进行了仿真实验，并与现有的求解组合优化问题的 QPSO 算法、DPSO 算法和 GPSO 算法进行比较。随机生成 100～500 个顶点的 15 个测试图作为测试用例。实验仿真结果表明，EDPSO 处理 15 个测试图得到的最大值和平均值比量子 PSO（QPSO）算法、DPSO 算法和 GPSO 算法的都好，甚至 EDPSO 求解每一个测试图所得的平均值都要优于 QPSO 算法求解对应测试图所得的最大值。仿真结果表明新算法具有非常好的寻优性能。

四种算法 20 次独立实验的平均运行时间结果表明 DPSO 算法的运行速度最慢，原因是它在每一次迭代时需要花费时间计算 Sigmoid 函数。EDPSO 算法的运算速度稍慢于 QPSO 算法和 GPSO 算法，因为该算法在每一次迭代中需要花费时间更新概率分布向量，但增加的时间开销很小。所以，EDPSO 的总体性能要优于 QPSO 算法、DPSO 算法和 GPSO 算法。

2．模糊离散PSO算法

庞巍等基于模糊思想，提出了一种改进的 PSO 算法，用于求解旅行商问题[5]。这种算法中采用模糊矩阵来表示粒子的位置和速度，并重新定义其更新公式。算法的主要步骤如下。

Begin

按照下列方法初始化粒子的位置和速度

$$\boldsymbol{X}=\begin{bmatrix} x_{11} & \mathrm{L} & x_{1n} \\ \mathrm{M} & \mathrm{O} & \mathrm{M} \\ x_{n1} & \mathrm{L} & x_{nn} \end{bmatrix}$$

矩阵中的元素按照如下的条件随机产生:

（1）$\sum_{j=1}^{n} x_{ij}=1,\ i=1,2\mathrm{L}\ ,n$;

（2）$x_{ij}\in(0,1)$。

速度的初始化:

$$\boldsymbol{V}(0)=\begin{bmatrix} v_{11} & \mathrm{L} & v_{1n} \\ \mathrm{M} & \mathrm{O} & \mathrm{M} \\ v_{n1} & \mathrm{L} & v_{nn} \end{bmatrix}$$

矩阵中的元素也是随机产生的，并且满足如下条件:

$$\sum_{j=1}^{n} v_{ij}=0,\qquad i=1,2,\mathrm{L}\ ,n$$

While（终止准则不满足）do

（1）按照下列各式更新粒子的速度和位置，其中⊕和⊗分别表示矩阵的乘法和加法运算:

$$\boldsymbol{V}_i\ (t+1)=w\otimes\boldsymbol{V}_i\ (t)\oplus[c_1\mathrm{grand}()\otimes(\boldsymbol{P}_i-\boldsymbol{X}_i)\oplus[c_2\mathrm{grand}()]\otimes(\boldsymbol{P}_g-\boldsymbol{X}_i)$$

$$\boldsymbol{X}_i(t+1)=\boldsymbol{X}_i(t)\oplus\boldsymbol{V}_i(t+1)$$

（2）归一化处理。经过一定次数的迭代后，位置矩阵 $x_{ij}\in(0,1)$ 的条件可能不成立，因此有必要对其进行归一化。首先将矩阵中所有的负数清零，然后将位置矩阵在满足 $\sum_{j=1}^{n} x_{ij}=1\ \ (i=1,2\mathrm{L}\ ,n)$ 的情况下进行如下的变换:

$$\begin{bmatrix} x_{11}/\sum_{i=1}^{n} p_{1i} & x_{12}/\sum_{i=1}^{n} p_{1i} & \text{L} & x_{1n}/\sum_{i=1}^{n} p_{1i} \\ \text{M} & \text{M} & \text{M} & \text{M} \\ x_{n1}/\sum_{i=1}^{n} p_{ni} & x_{n2}/\sum_{i=1}^{n} p_{ni} & \text{L} & x_{nn}/\sum_{i=1}^{n} p_{ni} \end{bmatrix}$$

（3）对新位置进行非模糊化，计算新位置的适应度值。如果新位置的适应度值优于当前局部最好解，则用新的位置更新当前的局部最好解。如果某个粒子的局部最好解优于当前的全局最好解，则用此局部最好解更新当前的全局最好解。

End While

输出全局最好解及其数据。

End

实验结果表明了算法的有效性，但同时也指出该算法没有专门针对 TSP 问题的经典算法（如 Lin-Kernighan 算法）那么高效。

3．免疫离散PSO算法

叶东毅等将疫苗接种和免疫选择等免疫机制引入到 BPSO 算法中，以基于决策表差别矩阵的某种属性重要性度量作为疫苗模式，提出了一种基于免疫粒子群优化思想的最小属性约简算法，称为免疫离散 PSO 算法（IPSO 算法）[6]。IPSO 算法的主流程如下所示。

Begin

初始化粒子群，计算问题决策表的差别矩阵 $\boldsymbol{M}$;

抽取疫苗。

（1）如果 $Core(C) \neq \phi$，假设 $Core(C) = \{a_{i1}, \text{L}, a_{is}\}$，$s < m$，并记 $I = \{i_1, \text{L}, i_s\}$；

（2）对 $\forall a \in C / Core(C)$，计算 a 在差别矩阵 $\boldsymbol{M}$ 元素中出现的频率 $f_{g(a)}$，

计算映射

$$h(a) = 0.8 \times \frac{f_{q(a)} - f_{q\min}}{f_{q\max} - f_{q\min}} + 0.1$$

其中，$f_{q\max}$ 和 $f_{q\min}$ 分别为所有频率 $f_{q(a)}$ 中的最大值和最小值。显然，

$h(a) \in [0.1, 0.9]$。

（3）按顺序确定疫苗位模式如下:

$$H_C = \{h(a_j) \mid j \in \{1, \text{L}\ , m\} \mid I\}$$

While（终止准则不满足）do

（1）根据适应度函数，计算当前粒子群中各个粒子的适应度值，得到个体历史最优和群体适应度值最高的全局最优粒子;

（2）更新粒子（抗体）群:

① 对粒子群中的粒子按照其适应度值由大到小排序，设前 k 个适应度值最高的粒子为$\{x_{p1}(t), \cdots, x_{pk}(t)\}$，记 $KM(t)=\{x_1=(t), \cdots x_p(t)\}$;;

② 根据基本 BPSO 算法，更新粒子的位置和速度。

③ 采用 k-精英策略：如果适应度值 $f(x_i(t+1)) < f(x_i(t(1))$，$x_i(t) \in KM$，则令

$$x_i(t+1)) = x_i(t)$$

④ 疫苗接种和免疫选择。

在粒子群中进行 q 次随机抽取，$IM(t+1) = \{x_{j1}(t+1), \text{L}\ , x_{jq}(t+1)\}$ 为抽取出的粒子集合，按如下方式进行疫苗接种和选择.

对每个 $x(t+1) \in IM(t+1)$，令

$$Bound_j = \begin{cases} h(a_j) - 0.1 & x_j(t+1) = 0 \\ h(a_j) + 0.1 & x_j(t+1) = 1 \end{cases}, \quad j \in \{1, \text{L}\ , m\} / I$$

接种疫苗得到一个新粒子

$$Ix_j(t+1) = \begin{cases} 1 & Bound_j > rand \\ 0 & Bound_j \leqslant rand \end{cases}, \quad j \in \{1, \text{L}\ , m\} / I$$

如果适应度值 $f(x(t+1)) > f(Ix(t+1))$ 则取消对该粒子(抗体)的接种；否则，$x(t+1)) = Ix(t)$。

End While

输出全局最优及其他数据。

End

为了测试（IPSO）算法的有效性，作者用 UCI 数据库中的 5 个数据集作为算例进行计算，并同基于可行域的遗传约简算法（简记为 GA1）、启发式遗传算法（简记为 GA2）、基于 BPSO 的带奖励因子的约简算法（简记为 PS1）以及基于修正算子的粒子群优化约简算法（简记为 PS2）等 4 个基于全局优化方法的最小属性约简算法进行比较，结果见表 3.1 和表 3.2。每个算法都独立运行 10 次，直到达到停机准则为止。判断每个算法 10 次运行求得的属性子集中是否包含属性约简，若有，则取其中最小的一个，并记 N_R 为其包含的属性个数；若没有，则取 N_R=Null。C_B 为每个算法执行 10 次中获得一个最小属性约简的次数，它表示该算法的成功率；t_m 为 10 次所用的平均计算时间，单位为 s。对缺失的数据采用取 Null 值的处理方法。

表 3.1　计算结果和成功率比较（N_R/C_B）

数据集	PS1	PS2	GA1	GA2	IPSO
Zoo	5/2	5/5	5/3	5/5	5/7
Soybean-Large	12/0	10/0	13/0	12/0	9/3
House	4/2	4/3	4/2	4/2	4/6
Lympho-graphy	10/0	9/0	10/0	10/0	8/3
Sponge	10/0	9/0	10/0	10/0	8/4

表 3.2　平均计算时间比较（t_m）　　单位：s

数据集	PS1	PS2	GA1	GA2	IPSO
Zoo	32	25	37	31	13
Soybean-large	1431	1302	2136	1727	667
House	71	67	52	47	21
Lympho-graphy	1390	1299	1401	1362	515
Sponge	1756	1654	1783	1933	752

从上述计算结果可以看出，相对其他 4 个算法的最小属性约简算法而言，无论是在解的质量方面还是在计算时间开销上，IPSO 算法均有较为明显的优势， 这主要得益于适应度值函数的较为合理的定义以及差别矩阵信息的利用。

4．离散量子PSO算法

2003 年，Jun Sun 等在分析 PSO 算法的基础上，将量子行为引入了粒子群算法，

这样更加符合社会智能群体的进化模型，提出了量子 PSO（Quantum PSO，QPSO）算法[7]，算法方程如下：

$$m_{\text{best}} = [m_1(t), m_2(t), \text{L}\ , m_d(t)] = [\frac{1}{N}\sum_{i=1}^{N} p_{i1}(t), \frac{1}{N}\sum_{i=1}^{N} p_{i2}(t), \text{L}\ , \frac{1}{N}\sum_{i=1}^{N} p_{id}(t)] \tag{3.8}$$

$$\boldsymbol{P} = (\phi_1 * \boldsymbol{P}_i + \phi_2 * \boldsymbol{P}_g) / (\phi_1 + \phi_2) \tag{3.9}$$

$$\boldsymbol{X}_i(t+1) = P \pm \beta \left| m_{\text{best}} - \boldsymbol{X}_i(t) \right| * \ln(1/u) \tag{3.10}$$

其中，ϕ_1、ϕ_2 是在[0，1]之间产生的随机数，m_{best} 称为中值最优位置，β是创造力系数，$\boldsymbol{X}_i(t)$ 为粒子 i 在 t 次迭代过程中的相关位置信息，N 是群体中所含粒子的数目，u 是[0，1]之间的随机数。在每次迭代过程中，加、减算法是由（0，1）之间随机产生的随机数的大小决定的，当产生的随机数大于 0.5 时，取减号，其他情况取加号。QPSO 算法在函数测试、滤波器设计、多阶段金融规划、神经网络优化和 $H\infty$ 控制等许多方面得到应用，并取得了较好的仿真效果。

2004 年，Yang 等基于 QPSO 算法提出了离散 QPSO（Discrete QPSO）算法[8]，QPSO 算法的粒子群表述为：

$$\boldsymbol{X} = [\boldsymbol{X}_1, \boldsymbol{X}_2, \text{L}\ , \boldsymbol{X}_M] \tag{3.11}$$

其中，下标 M 为粒子群的种群规模。

种群中的第 i 个粒子表示为

$$\boldsymbol{X}_i = [x_{i1}, x_{i2}, \text{L}\ , x_{iN}] \tag{3.12}$$

其中，下标 N 为粒子离散化后的位数。

离散粒子每一位只可取 0 或 1。算法的更新公式如下：

$$v_{ipbest,j}(t+1) = \alpha \cdot p_{ij}(t) + \beta \cdot (1 - p_{ij}(t)) \tag{3.13}$$

$$v_{gbest,j}(t+1) = \alpha \cdot p_{gj}(t) + \beta \cdot (1 - p_{gj}(t)) \tag{3.14}$$

$$x_{ij}(t+1) = w \cdot x_{ij}(t) + c_1 \cdot v_{ipbest,j}(t) + c_2 \cdot v_{gbest,j}(t) \tag{3.15}$$

$$x_{ij}(t+1)=\begin{cases}1 & \text{rand}() > v_{ij}(t+1)\\ 0 & \text{rand}() \leqslant v_{ij}(t+1)\end{cases} \tag{3.16}$$

其中，rand() 为分布在[0，1]范围内随机数；α、β（$0<\alpha$，$\beta<1$）为控制参数，代表了算法对速度的控制。其他参数同基本 PSO 算法，且参数满足：$0<w,c_1,c_2<1$，$w+c_1+c_2=1$。具体的算法流程如下。

```
Begin
  初始化离散粒子群的速度和位置;
  计算粒子的适应度值，初始化全局最优和各粒子的局部最优。
  While（终止准则不满足）do
    根据更新公式。更新粒子的速度的位置;
    计算粒子的适应度值，更新全局最优和各粒子的局部最优。
  End While
  输出全局最优及其他数据。
End
```

2006 年 Chen[9]等用离散量子优化算法来求解 CVRP 问题，并与遗传算法（GA）和模拟退火算法（SA）在同一问题的运算结果作了对比。每种算法对每个函数进行五次搜索运算，取最优一次的结果作为对比参考量，如表（所示）。

表中，n 表示 CVRP 问题中客户个数，m 表示车辆数目。从比较的结果可以看出，离散量子粒子群算法的搜索性能是最好的。

3.2　小生境PSO算法

普通的进化算法（无论是原始的 PSO 算法还是 GA 算法）对于一个优化问题只能发现一个最优解。当面临一个多模优化问题需要发现多个最优解时，普通的进化计算技术将无能为力。即使将普通进化算法多次使用，也不能保证所发现的最优解互不相同。

在自然界“物以类聚，人以群分”是一种司空见惯的现象，生物总是倾向于与自己特征、性状相类似的生物生活在一起，一般总是与同类交配繁衍后代。在生物学中，把某种特定环境及其在此环境中生存的组织（Organism）称为小生境（Niche），而把有共同特性的一些组织称作物种（Species）。

小生境技术源于遗传算法，是遗传算法求解多峰函数时模拟生物小生境组织功能的一种技术。这种方法能使普通进化算法具有发现多个最优解的能力。在遗传算法中，发现多模问题中多个最优解的能力被称为小生境技术（Niching Technique）或

称为物种技术（Speciation Techniques）。自 DeJong 在 1975 年提出基于排挤机制的小生境选择策略后，有多种成小生境的技术被提出，如适应度值共享小生境、预选择机制策略、排挤机制等。

R. Brits 等人在 PSO 算法中引入了小生境技术，提出了基于 PSO 算法的小生境 PSO 算法[10]。该算法中，为了保持粒子群的多样性，若某个粒子在连续多次的迭代过程中其适应度值变化量很小，则以此粒子为中心，以该粒子与最近粒子的距离为半径构造一个圆形“小生境”。小生境的子粒子群的半径定义为：

$$R_{s_j} = \max\{\left\| x_{s_j,g} - x_{s_j,i} \right\|\} \tag{3.17}$$

其中，$x_{s_j,g}$、$x_{s_j,i}$ 分别为粒子群 S_j 中的最优粒子和任意的一非最优粒子。算法有两个主要的操作。

（1）若粒子 x_i 进入子群 S_j 范围内，即 $\left\| x_i - x_{s_j,i} \right\| \leqslant R_{s_j}$，则粒子将被此小生境子粒子群吸收。

（2）若两个子群 S_j、S_k 范围相交，即 $\left\| x_{s_j,g} - x_{s_k,g} \right\| \leqslant \left| R_{s_j} - R_{s_k} \right|$，则两个子群将被合并成一个子群。

与小生境遗传算法类似，R. Brits 等提出的小生境 PSO 算法也具有并行特点，通过基于适应度值的判断来发现和跟踪目标函数的多个最优解：首先采用仅有认知模式的 PSO 算法在主粒子群中搜索，当发现个体粒子的适应度值符合小生境生成条件时，则在该粒子的拓扑邻域内形成小生境子粒子群；小生境子粒子群根据保证收敛粒子群子优化（Guaranteed Convergence Particle Swarm Optimization，GCPSO）算法寻优。GCPSO 算法是 F. Van Den Bergh 于 2002 年提出，下面对 GCPSO 算法进行讲述[11]。

设 τ 为当前全局最优粒子的编号，为保证当前全局最优粒子在到达一个局部最优解位置之前能够保持运动，F. Van Den Bergh 提出的当前全局最优粒子速度更新公式为

$$v_{\tau j}(t+1) = -x_{\tau j}(t) + p_{gj}(t) + wv_{\tau j}(t) + \rho(t)(1 - 2r_{2j}(t)) \tag{3.18}$$

将式（3.18）代入标准 PSO 算法的位置更新公式，则可以得到全局最优粒子 τ 的位置更新公式

$$x_{\tau j}(t+1) = p_{gj}(t) + wv_{\tau j} + \rho(t)(1-2r_{2j}(t)) \tag{3.19}$$

算法中其他粒子仍然按照标准 PSO 算法的速度和位置更新公式。

式（3.18）中，$-x_{\tau j}(t)$项将粒子的位置重置到 $P_{g,j}$，$wv_{\tau,j}(t)$ 代表了当前最优粒子从该位置上开始的搜索方向；而 $\rho(t)(1-2r_{2,j}(t))$ 项则在 $P_{g,j}$ 周围大小为 $2\rho(t)$ 的样本空间内产生一个随机搜索，搜索半径由尺度因子 $\rho(t)$ 决定。$\rho(t)$ 的值随迭代次数 t 更新如下

$$\rho(t+1) = \begin{cases} 2\rho(t) & n_s > S_c \\ 0.5\rho(t) & n_f > f_c \\ \rho(t) & \text{其他} \end{cases} \tag{3.20}$$

其中，*#successes* 和 *#fallures* 分别表示搜索连续成功或失败的次数，当 $f(P_{g,j}(t)) = f(P_{g,j}(t+1))$ 时即认为搜索失败一次。$\rho(t)$ 的值按式（3.20）迭代的过程中还需以下两条规则

$$\begin{aligned} n_s(t+1) > n_s(t) &\Rightarrow n_f(t+1) = 0 \\ n_f(t+1) > n_f(t) &\Rightarrow n_s(t+1) = 0 \end{aligned} \tag{3.21}$$

即搜索成功时 n_f被置零，而搜索失败时 n_s 被置零。式（3.20）中还有两个优化参数 S_c 和 f_c，文献[11]建议 $S_c = 15$，$f_c = 5$，也可以对其进行动态学习获得。小生境 PSO 算法的迭代流程如下：

Begin
　初始化主粒子群。
　While（终止准则不满足）do
　　用认知模式的 PSO 对主粒子群进行一次优化迭代;
　　更新主粒子群中每个粒子的适应度值。
　　对每一个子粒子群:
　　（1）用 GCPSO 对子粒子群进行一次优化迭代;
　　（2）更新每一个粒子的适应度值;
　　（3）更新子粒子群的半径;
　　如果子粒子群满足合并条件，合并子粒子群;
　　如果主粒子群中的微粒进入子粒子群半径范围，则被该子粒子群吸收;
　　检查主粒子群中是否有粒子满足小生境产生条件，若满足则该粒子及其邻域中的粒子形成子粒子群。

```
    End While
End
```

参考文献[12]用小生境算法对 f_1～f_5 五个多峰测试函数进行了寻优测试，采用的参数和测试结果分别见表 3.3 和表 3.4。

表 3.3　算法参数选择

参数 / 函数	δ	μ	S	$x_{\min}$	$x_{\min}=v_{\max}$
f_1	0.0001	0.001	30	0.01	1.0
f_2	0.0001	0.001	30	0.01	1.0
f_3	0.0001	0.001	30	0.01	1.0
f_4	0.0001	0.001	30	0.01	1.0
f_5	0.0001	0.001	30	-5.0	1.0
f_6	0.0001	0.001	50	0.0	50.0

表 3.4　测试结果

函　数	适 应 值	偏　差	收 敛 率
f_1	7.68×10^{-5}	2.20×10^{-4}	100%
f_2	9.12×10^{-2}	6.43×10^{-2}	100%
f_3	5.95×10^{-6}	4.86×10^{-5}	100%
f_4	8.07×10^{-2}	6.68×10^{-2}	100%
f_5	4.78×10^{-6}	1.03×10^{-5}	100%
f_6	1.778×10^{-1}	6.36×10^{-4}	20%

从测试结果来看，小生境 PSO 算法能够保持粒子群的多样性，成功地发现 f_1～f_5 这样的多峰函数的每一个峰值，显示了比较好的性能。

由于小生境 PSO 算法的小生境产生依赖于一个圆形的拓扑领域，不能满足 Schaffer 函数等一些峰为带状或环状的多峰函数的优化要求，且与生物学中地理小生境具有多种形状的事实也不符。王俊年等将“基于密度”的聚类算法引入小生境 PSO 算法,构建了一种聚类小生境 PSO 算法（Clustering Based Niching PSO，CBNPSO）[13]。该算法能够产生不同形状和大小的小生境，弥补了小生境算法的不足。CBNPSO 算

法组合了两种方法来实现技术的思想：第一，采用多种群策略实现全局和各子微粒群按不同的 PSO 算法进化；第二，采用聚类算法区分粒子群中存在的不同子粒子群。

为测试 CBNPSO 算法的性能，这里选择了表 3.5 所示的 3 个典型的多峰函数对该算法和小生境 PSO 算法、基本 PSO 算法进行了对比测试。测试 f_1、f_2、f_3 时初始化粒子群规模分别为 40、30、60，算法的其他参数选择见表 3.6，分别运行 300 代后，测试性能见表 3.7 所示。

表 3.5　测试函数

$f_1 = e^{\frac{-x}{T}(1-\sin x)}$，$0 \leqslant x \leqslant 20\pi$，$T=80$
$f_2 = (x^2 - \cos 20\pi x + 10) + (y^2 - \cos 20\pi y + 10)$，$-5.12 \leqslant x$，$y \leqslant 5.12$
$f_3 = \dfrac{\sin(\sqrt{(x-25)^2 + (y-25)^2} + 2718}{\sqrt{(x-25)^2 + (y-25)^2} + 2718}$，$0 \leqslant x$，$y \leqslant 50$

表 3.6　测试参数

参数 / 算法	c_1	c_2	W	c_{n1}	c_{n2}	w_n	δ_{dist}		
							f_1	f_2	f_1
基本 PSO	1.5	0	0.8	—	—	—	—	—	—
小生境 PSO	1.5	0	0.8	1.3	1.3	0.8	—	—	—
CBNPSO	1.5	0	0.8	1.3	1.3	0.8	1.0	0.8	1.5

表 3.7　测试性能

函数	基本 PSO			小生境 PSO			CBNPSO		
	找到全局最大（小）值		找到全部峰谷（次数）	找到全局最大（小）值		找到全部峰谷（次数）	找到全局最大（小）值		找到全部峰谷（次数）
	次数	平均值		次数	平均值		次数	平均值	
f_1	8	3.7741	7	20	4.3019	19	20	4.3019	17
f_1	7	0.1820	7	19	0.05646	20	19	0.05333	20
f_1	7	-0.15165	3	20	-0.17780	0	20	-0.1778	18

从表 3.7 的实验结果可以看到，基本 PSO 算法在各种多峰函数寻优中效率很低，尤其是像 f_3 这样的特殊函数，基本 PSO 算法找到全局最优解和所有局部最优解的概率都不超过 20%；小生境 PSO 算法在 f_1、f_2 这样的一般多峰函数寻优中表现突出，

几乎每一次都能够找到全局最优解和每一个局部最优解，而 CBNPSO 算法不仅在一般多峰函数中具有和小生境 PSO 算法同样的寻优能力，而且在 f_3 这样的特殊多峰函数寻优中，展现了良好的性能。

贾东立等结合小生境策略全局优化与变尺度混沌变异精细搜索各自的优点，提出基于混沌变异的小生境 PSO 算法，并在算法中引入了种群淘汰策略，结合小生境的子种群竞争策略一起使用[14]。算法迭代过程首先利用 RCS 竞争策略，使各个小生境子种群形成独立的搜索空间，追逐不同的极值点；然后每隔一定代数，对陷入局部最优的最劣子种群进行随机初始化。这样可使种群在不断的竞争和更新中向前进化，从而避免了算法早熟收敛，保证了收敛到全局最优。而没有更新的小生境种群继续向前进化，又保证了搜索精度的连续提高。

为了验证小生境 PSO 算法对复杂高维函数的有效性，采用高维基准测试函数进行测试。比较了小生境 PSO 算法与基本 PSO 算法、CPSO 算法和 PSCO 算法对基准测试函数的效果。算法运行代数统一设为 2 000，2 000 代后认为算法陷入停滞。表 3.8 为用到的基准测试函数，表 3.9 为 50 次独立运行的测试结果。从表 3.9 可以看出，小生境 PSO 算法能解决其他算法难以解决的高维多极函数的优化问题，并具有更强的寻优能力和较高的搜索精度。

表 3.8　高维基准测试函数

函　　数	表　达　式
Rosenbrock 函数	$f_1(x)=\sum_{i=1}^{n-1}(100(x_{i+1}-x_i^2)^2+(x-1_i)^2), x_i\in[-30,30], n=30$
Rastrigin 函数	$f_2(x)=\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)+10, x_i\in[-5.12,5.12], n=30$
Ackley 函数	$f_3(x)=-20\mathrm{e}^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}}-\mathrm{e}^{\frac{1}{n}\sum_{i=1}^{n}\cos(20\pi x_i)}+20+e$ $x_i\in[-5.12,5.12], n=30$

表 3.9　高维复杂函数优化测试结果

函　　数	算　　法	均　　值	标准误差	R_s（%）
Rosenbrock 函数	小生境 PSO	0.261	0.313	96
	CPSO	1.741	2.858	50
	PSCO	18.184	5.364	0
	PSO	85.514	35.63	0

续表

函　　数	算　　法	均　　值	标准误差	R_s（%）
Rastrigin 函数	小生境 PSO	0.15×10^{-4}	0.23×10^{-4}	100
	CPSO	6.462	8.343	32
	PSCO	9.580	10.220	8
	PSO	25.810	6.946	0
Ackley 函数	小生境 PSO	0.31×10^{-6}	0.16×10^{-5}	100
	CPSO	0.262	0.682	88
	PSCO	0.502	0.431	92
	PSO	0.452	0.594	68

3.3　混合PSO群算法

针对 PSO 算法存在的缺陷，研究人员将其他进化算法或传统优化算法与 PSO 算法进行混合，用于提高种群中粒子的多样性，平衡粒子群的全局探索能力和局部开发能力，改善粒子群的收敛性能和精度。用于这种结合的途径通常有两种：

（1）将 PSO 与其他进化算法结合，利用进化算法中的变异、选择和交叉操作来保留种群中性能最好的粒子。

（2）利用其他优化技术自适应调整收缩因子/惯性权值、加速常数等。

遗传算法（GA）、模拟退火算法（SA）及差分进化算法（DE）通常用来和 PSO 算法进行结合。通过利用这些进化算法中的交叉操作，两个不同的粒子可以互相交换信息，增加了粒子“飞行”到搜索空间其他新位置的能力；通过利用变异操作，可以增加种群的多样性，可以避免粒子群陷入局部最优。

3.3.1　PSO-DV算法

Das 等人在基本 PSO 中引入差分算子（Differential Operator）来更新种群中每个粒子的速度，在这种算法中，速度更新公式中反映粒子自我认知的第二部分被种群中另外两个粒子权重化了的位置矢量的差所代替。由于这种算法是受差分进化算法中变异思想的启发，因此 Das 命名这种算法为 PSO-DV 算法。同时，这个算法还根据 DE 算法的贪婪选择原理引入了“适者生存”的机制[15]。

PSO-DV 算法中，对于粒子 i，可以随机选择另外的两个粒子 j 和 k（$i \neq j \neq k$），利用它们的位置坐标的差构成一个微分微量，表示为：

$$\boldsymbol{\delta} = \boldsymbol{X}_k - \boldsymbol{X}_j \tag{3.22}$$

则第 i 个粒子的速度矢量的第 d 个元素可以用下式更新：

$$\begin{cases} \boldsymbol{V}_{id}(t+1) = \omega \cdot \boldsymbol{V}_{id}(t) + \beta \cdot \delta_d + C_2 \cdot \varphi_2 \cdot (P_{gd} - jX_{id}) \ , (\text{rand}_d(0,1) \leqslant C_R) \\ \boldsymbol{V}_{id}(t+1) = V_{id}(t), \qquad\qquad \text{其他} \end{cases} \tag{3.23}$$

其中，C_R 是交叉概率，δ_j 是定义的微分向量的第 j 个元素，β是[0，1]上的一个比例因子。本质上来说，速度更新公式中“认识”部分被差分算子取代可以导致额外的探索能力。很明显，如果 $C_R \leqslant 1$，速度向量中的一些元素保留它们的旧值。现在，通过旧位置 $\boldsymbol{X}_i$ 以更新的速度得到一个新的试验位置 T_{ri}：

$$\boldsymbol{T}_{ri} = \boldsymbol{X}_i(t) + \boldsymbol{V}_i(t+1) \tag{3.24}$$

如果新的位置的适应度值更好，则粒子的位置被新的位置取代。因此，当搜索一个 n 维函数 $f(\boldsymbol{X})$ 的最小值时，目标粒子的位置按照下式更新：

$$\begin{cases} \boldsymbol{X}_i(t+1) = \boldsymbol{T}_{ri}, \qquad f(\boldsymbol{T}_{ri}) \leqslant f(\boldsymbol{X}_{ri}) \\ \boldsymbol{X}_i(t+1) = \boldsymbol{X}_i(t), \quad \text{其他} \end{cases} \tag{3.25}$$

因此，它的速度每次被更新，粒子可以移动到搜索空间的更好位置，也可以保持先前的位置。因此，粒子的当前位置是它曾经发现的最好位置。换句话说，和基本 PSO 算法不同，在当前的方法中，局部最优总是等于当前位置向量。因此，包含 $\left|p_{ij} - x_{ij}\right|$ 的“认识”部分自动排除。如果一个粒子在搜索空间变得停滞（即它的位置在迭代过程中不再改变），粒子的位置会通过随机的变异被改变到一个新的位置。这种“技巧”可以避免粒子陷入局部最优，帮助粒子移动。

If　　$\boldsymbol{X}_i(t) = \boldsymbol{X}_i(t+1) = \boldsymbol{X}_i(t+2) = \text{L}\ \boldsymbol{X}_i(t+N)$, 且 $f\ [\boldsymbol{X}_i(t+N)] = f^*$

then

for($r = 1$ to n)

$$\boldsymbol{X}_i(t+N+1) = \boldsymbol{X}_{\min} + \text{rand}(0,1) \cdot (\boldsymbol{X}_{\max} - \boldsymbol{X}_{\min}) \tag{3.26}$$

其中，f^* 是适应度函数的全局最小值，N 是所能容忍的最大停滞步数，（$\boldsymbol{X}_{\max}$，$\boldsymbol{X}_{\min}$）是搜索空间可能的边界。算法流程如下所示。

```
Begin
    随机初始化粒子群。
    While（终止准则不满足）do
      For i=1 to n
```

```
            计算每个粒子的适应度值;
            更新全局最优;
            随机选择另外的两个粒子 j 和 k（ i ≠ j ≠ k ）按式（3.22）构造
            差分量;
            For d=1 至粒子维数
              按式（3.23）更新粒子的速度。
            End For
            根据式（3.24）生成试验位置向量;
            根据式（3.25）更新粒子的位置。
          End For
          For i=1 to n
            如果粒子停滞，根据式（3.26）更新停滞粒子。
          End For
        End While
End
```

Das 等人为了验证 PSO-DV 算法的性能，将算法对表 3.10 所示的 6 个基准函数进行了测试，并和其他的算法进行了比较，比较的结果见表 3.11。

表 3.10　基准测试函数

函数	表达式
Sphere 函数，f_1	$f_1(x)=\sum_{i=1}^{n}x_i^2$
Rosenbrock 函数，f_2	$f_2(x)=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2$
Rastrigin 函数，f_3	$f_3(x)=\sum_{i=1}^{n-1}[x_i^2-10\cos(2\pi x_i)+10]$
Griewank 函数，f_4	$f_4(x)=\frac{1}{4000}\sum_{i=1}^{n-1}x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$
Ackley 函数，f_5	$f_5(x)=-20\mathrm{e}^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}}-\mathrm{e}^{\frac{1}{n}\sum_{i=1}^{n}\cos(20\pi x_i)}+20+e$

续表

函数	表达式
Shekel's Foxholes 函数，f_6	$f_6(x)=[\frac{1}{500}+\sum_{j=}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}]^{-1}$

表 3.11　各种算法的平均值（标准偏差）比较

函数	D_{im}	N	$G_{\max}$	均值（标准偏差）					
				PSO	PSO-TVIW	MPSO-TVAC	HPSO	DE	PSO-DV
f_1	10	50	1000	0.001	0.001	0.001	0.001	0.001	0.001
	20	100	2000	0.001	0.001	0.001	0.001	0.001	0.001
	30	150	4500	0.001	0.001	0.001	0.0001	0.001	0.001
f_2	10	75	3000	21.705 (40.162)	16.21 (14.917)	1.234 (4.3232	1.921 (4.330)	2.263 (4.487)	0.0063 (0.0561)
	20	150	4000	52.21 (148.32)	42.73 (72.612)	22.732 (30.638)	20.749 (12.775)	18.934 (9.453)	0.0187 (0.554)
	30	250	5000	77.61 (81.172)	61.78 (48.933)	34.229 (36.424)	10.414 (44.845)	6.876 (1.688)	0.0227 (0.182)
f_3	10	50	3000	2.334 (2.297)	2.1184 (1.563)	1.78 (2.793)	0.039 (0.061)	0.006 (0.0091)	0.0014 (0.0039)
	20	100	4000	13.812 (3.491)	16.36 (4.418)	11.131 (0.91)	0.2351 (0.1261)	0.0053 (0.0032)	0.0028 (0.0017)
	30	150	5000	6.652 (21.811)	24.346 (6.317)	50.065 (21.139)	1.903 (0.894)	0.099 (0.112)	0.0016 (0.277)
f_4	10	50	2500	0.1613 (0.097)	0.092 (0.021)	0.00561 (0.047)	0.057 (0.045)	0.054 (0.0287)	0.024 (0.180)
	20	100	3500	0.2583 (0.1232)	0.1212 (0.5234)	0.0348 (0.127)	0.018 (0.0053)	0.019 (0.0113)	0.0032 (0.0343)
	30	150	5000	0.0678 (0.236)	0.1486 (0.124)	0.0169 (0.116)	0.023 (0.0045)	0.005 (0.0035)	0.0016 (0.0022)

续表

函数	D_{im}	N	$G_{\max}$	均值（标准偏差）					
				PSO	PSO-TVIW	MPSO-TVAC	HPSO	DE	PSO-DV
f_5	10	50	2500	0.406 (1.422)	0.238 (1.812)	0.169 (0.772)	0.0926 (0.0142)	0.00312 (0.0154)	0.00417 (0.1032)
	20	100	3500	0.572 (3.094)	0.318 (1.118)	0.537 (0.2301)	0.117 (0.025)	0.029 (0.0067)	0.0018 (0.028)
	30	150	5000	1.898 (2.598)	0.632 (2.0651)	0.369 (2.735)	0.068 (0.014)	0.0078 (0.0085)	0.0016 (0.0078)
f_6	2	40	1000	1.235 (2.215)	1.239 (1.468)	1.321 (2.581)	1.328 (1.452)	1.032 (0.074)	0.9991 (0.0002)

3.3.2　GA-PSO算法

遗传算法（GA）是基于自然选择、进化和遗传的随机搜索算法。这类算法同时搜索解空间的多个点，因此容易找到所给问题的全局最优解；同时遗传算法直接以适应度作为搜索信息，无需导数等其他辅助信息，容易实现。遗传算法的缺点在于，迭代过程中种群发生改变时，会丧失以前的信息。而 PSO 算法具有记忆能力，能够保留局部个体和全局种群的最优信息。Yi-Tung Kao[16]等结合了遗传算法和 PSO 算法的优点，提出了遗传算法和 PSO 算法相混合的算法，即 GA-PSO 算法，其原理如图 3.7 所示。

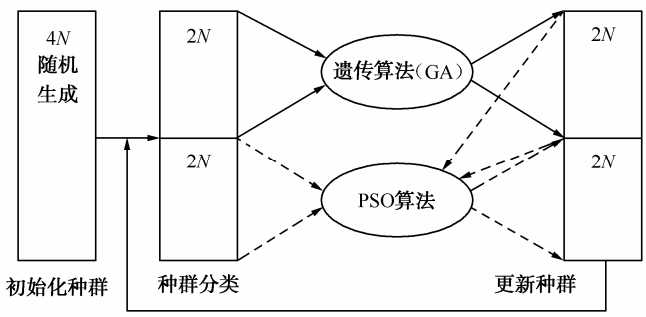

图 3.7　GA-PSO 算法原理图

在 GA-PSO 算法中，如果问题的解空间是 N 维的，则种群中被随机初始化为 $4N$ 个个体。这些个体相当于遗传算法中的染色体和 PSO 算法中的粒子。这 $4N$ 维的个体按照适应度值排列，前 $2N$ 个个体被提供给实数编码的遗传算法，利用交叉和变异操作产生 $2N$ 个新个体。新产生的这 $2N$ 个个体通过 PSO 算法来调整剩余的 $2N$ 个个体。PSO 算法的调整过程包括全局最优粒子的选择、局部最优粒子的选择和速度的更新。在 GA-PSO 算法中，根据适应度值来选择全局最优；$2N$ 个个体平均分为 N 个领域，选择每个邻域中适应度值最好的个体作为局部最优。整个算法的流程如下所示。

Begin

对 N 维问题，随机产生 $4N$ 大小的种群。

While（终止准则不满足）do

评价并分类。根据适应度函数，计算 $4N$ 个个体的适应度值，并根据适应值把它们分为 2 组，每组 $2N$ 个;

应用 GA 算法产生新个体。利用实数编码的 GA 算法对适应度值最好的 2N 个个体通过交叉和变异操作，产生新的 $2N$ 个个体。

（1）选择。从种群中选择 $2N$ 个适应度值最好的个体。

（2）对选出的 $2N$ 个个体，以 100%的交叉概率，根据下列方程进行更新:

$$\boldsymbol{X}_i' = \text{Uniform}(0,1)\boldsymbol{X}_i + (1-\text{Uniform}(0,1))\boldsymbol{X}_{i+1},\quad i = 1,2,\text{L}\ ,2N-1$$

$$\boldsymbol{X}_i' = \text{Uniform}(0,1)\boldsymbol{X}_i + (1-\text{Uniform}(0,1))\boldsymbol{X}_1,\quad i = 2N$$

（3）以 20%的变异概率，对选出的 2N 个染色体根据下式进行变异操作:

$$\boldsymbol{X}_k' = \boldsymbol{X}_k + \text{rand}() \times N(0,1)$$

根据基本 PSO 算法的速度和位置更新公式，更新剩余的适应度值最坏的 $2N$ 个个体。

End While

输出全局最优。

End

为了测试 GA-PSO 算法的效果，Yang 对 benchmark 测试集中的 17 个函数连续运行 100 次，分析了变异概率对 GA-PSO 算法的影响，给出了成功率、平均评价次数、平均误差，见表 3.12，并给出了与 CGA 和 CHA 算法的比较结果。

表 3.12　CGA、CHA 和 GA-PSO 算法的测试结果

函数	成功率（%）			函数平均评价次数			平均公差		
	CGA	CHA	GA-PSO	CGA	CHA	GA-PSO	CGA	CHA	GA-PSO
f_1	100	100	100	620	295	8254	0.0001	0.0001	0.00009
f_2	100	100	100	1504	952	809	0.0010	0.0010	0.00003
f_3	100	100	100	410	259	25706	0.0010	0.0010	0.00012
f_4	100	100	100	430	132	174	0.0003	0.0000002	0.00001
f_5	100	100	100	575	345	96211	0.0050	0.0050	0.00007
f_6	100	100	100	960	459	140894	0.0040	0.0040	0.00064
f_7	100	100	100	620	215	95	0.000003	0.000003	0.00005
f_8	100	100	100	750	371	206	0.0002	0.0002	0.00004
f_9	100	100	100	582	492	2117	0.0050	0.0050	0.00020
f_{10}	76	85	100	610	698	529344	0.1400	0.0090	0.00014
f_{11}	83	85	100	680	620	56825	0.1200	0.0100	0.00015
f_{12}	81	85	100	650	650	43314	0.1500	0.0150	0.00012
f_{13}	100	100	100	3990	3290	1358064	0.1500	0.0180	0.00013
f_{14}	100	100	100	1350	950	398	0.0004	0.00006	0.00000
f_{15}	100	100	100	970	930	12568	0.0400	0.0080	0.00024
f_{16}	80	83	100	21563	14563	5319160	0.0200	0.0080	0.00005
f_{17}	100	100	100	6991	4291	872	0.000001	0.000001	0.00000

3.4　SA-PSO算法

模拟退火算法（SA 算法）来源于固体退火原理，最初的思想由 Metropolis 在 1953 年提出，Kirkpatrick 于 1983 年成功地将其应用在组合最优化问题中。将固体加温至充分高，再让其徐徐冷却；加温时，固体内部粒子随温升变为无序状，内能增大；而徐徐冷却时粒子渐趋有序，每个粒子温度都达到平衡态，最后在常温时达到基态，内能减为最小。根据 Metropolis 的准则，粒子在温度 T 时趋于平衡的概率为 $E-\Delta E/(kT)$，其中 E 为温度 T 时的内能，ΔE 为其改变量，k 为 Boltzmann 常数。用固体退火模拟组合优化问题，将内能 E 模拟为目标函数值 f，温度 T 演化成控制参数 t，即得到解组合优化问题的模拟退火算法：由初始解 i 和控制参数初值 t 开始，

对当前解重复“产生新解→计算目标函数差→接受或舍弃”的迭代，并逐步衰减 t 值，算法终止时的当前解即为所得近似最优解，这是基于蒙特卡罗迭代求解法的一种启发式随机搜索过程。退火过程由冷却进度表（Cooling Schedule）控制，包括控制参数的初值 t 及其衰减因子 Δt 、每个 t 值时的迭代次数 L 和停止条件 S 。

SA 算法可以分解为解空间、目标函数和初始解三个部分。SA 算法的基本流程如下。

Begin

初始化：初始温度 T，初始解状态 S，每个 T 值的迭代次数 L。

While（终止准则不满足）do

（1）产生新解 S' ；

（2）计算增量 $\Delta t' = C(S') - C(S)$，其中 $C(S)$为评价函数；

（3）若 $\Delta t' < 0$ 则接受 S' 作为新的当前解，否则以概率 $e^{-\frac{\Delta t'}{T}}$ 接受 S' 作为新的当前解；

（4）退火操作：T 逐渐减小，$T_{k+1} = CT_k$ ，$C \in (0,1)$ 。

End While

输出当前解最优解。

End

终止条件通常取为连续若干个新解都没有被接受时终止算法。在上面的流程中，模拟退火算法新解的产生和接受可分为如下四个步骤，对应了上面算法流程的（1）～（4）。

步骤（1）　由一个产生函数从当前解产生一个位于解空间的新解；为便于后续的计算和接受，减少算法耗时，通常选择由当前新解经过简单地变换即可产生新解的方法，如对构成新解的全部或部分元素进行置换、互换等，注意到产生新解的变换方法决定了当前新解的邻域结构，因而对冷却进度表的选取有一定的影响。

步骤（2）　计算与新解所对应的目标函数差。因为目标函数差仅由变换部分产生，所以目标函数差的计算最好按增量计算。事实表明，对大多数应用而言，这是计算目标函数差的最快方法。

步骤（3）　判断新解是否被接受,判断的依据是一个接受准则，最常用的接受准则是 Metropo1is 准则：若 $\Delta t' < 0$ 则接受 S' 作为新的当前解 S，否则以概率 $e^{-\frac{\Delta t'}{T}}$ 接受 S' 作为新的当前解 S。

步骤（4）　当新解被确定接受时，用新解代替当前解，这只需将当前解中对应于产生新解时的变换部分予以实现，同时修正目标函数值即可。此时，当前解实现

了一次迭代。可在此基础上开始下一轮试验。而当新解被判定为舍弃时，则在原当前解的基础上继续下一轮试验。

SA 算法与初始值无关，算法求得的解与初始解状态 S（是算法迭代的起点）无关；SA 算法具有渐近收敛性，已在理论上被证明是一种以概率 1 收敛于全局最优解的全局优化算法；SA 算法具有并行性。

2004 年高鹰[17]提出了基于模拟退火的 PSO 算法。该算法以基本 PSO 算法运算流程作为主体流程，为把模拟退火机制引入其中，采用杂交 PSO 算法中的杂交运算和带高斯变异的 PSO 算法中的变异运算，进一步调整优化群体。其基本的执行过程是先随机产生初始群体；开始随机搜索，通过基本 PSO 算法的速度和位置更新公式来产生一组新的个体；然后再独立地进行杂交运算和带高斯变异运算，通过对所产生出的各个个体分别进行模拟退火，以其结果作为下一代群体中的个体。

在每次进化中，杂交运算依据杂交概率选取指定数量的粒子放入一个池中。池中的粒子随机地两两杂交，产生同样数目的子粒子，并用子粒子代替父母粒子，以保持种群的粒子数目不变。子粒子的位置由父母粒子的位置的算术加权和计算，即

$$\text{child}_1(\boldsymbol{X}) = \boldsymbol{p} * \text{parent}_1(\boldsymbol{X}) + (1-\boldsymbol{p}) * \text{parent}_2(\boldsymbol{X}) \tag{3.28}$$

$$\text{child}_2(\boldsymbol{X}) = \boldsymbol{p} * \text{parent}_2(\boldsymbol{X}) + (1-\boldsymbol{p}) * \text{parent}_1(\boldsymbol{X}) \tag{3.29}$$

其中，$\boldsymbol{X}$ 是 d 维的位置向量，而 $\text{child}_k(\boldsymbol{X})$ 和 $\text{parent}_k(\boldsymbol{X})$，$k$=1,2，分别表示子粒子和父母粒子的位置；$\boldsymbol{p}$ 是 d 维均匀分布的随机数向量，$\boldsymbol{p}$ 的每个分量都在[0,1]取值。子粒子的速度分别由下面的公式得到：

$$\text{child}_1(\boldsymbol{V}) = \frac{\text{parent}_1(\boldsymbol{V}) + \text{parent}_2(\boldsymbol{V})}{\left|\text{parent}_1(\boldsymbol{V}) + \text{parent}_2(\boldsymbol{V})\right|} \cdot \left|\text{parent}_1(\boldsymbol{V})\right| \tag{3.30}$$

$$\text{child}_2(\boldsymbol{V}) = \frac{\text{parent}_1(\boldsymbol{V}) + \text{parent}_2(\boldsymbol{V})}{\left|\text{parent}_1(\boldsymbol{V}) + \text{parent}_2(\boldsymbol{V})\right|} \cdot \left|\text{parent}_2(\boldsymbol{V})\right| \tag{3.31}$$

其中，$\boldsymbol{V}$ 是 d 维的速度向量，而 $\text{child}_k(\boldsymbol{V})$ 和 $\text{parent}_k(\boldsymbol{V})$，$k$=1、2，分别表示子粒子和父母粒子的速度。

在每次进化中，变异运算依据变异概率选取指定数量的粒子按高斯变异算子进行变异，用变异后的粒子代替原粒子，即

$$\text{mutation}(\boldsymbol{X}) = \boldsymbol{X} * (1 + \text{Gaussian}(\delta)) \tag{3.32}$$

整个算法的执行过程由两部分组成，首先通过基本的粒子群优化算法的进化操作（侧重全局搜索）产生出较优良的一个群体，然后再应用杂交运算和变异运算在

模拟退火操作（侧重局部搜索）下来进行粒子的进一步优化调整。进化过程反复迭代，直到满足某个终止条件为止，其算法流程如下。

开始

步骤 1：初始化交叉概率 P_c、变异概率 P_m、温度冷却系数 C、退火初始温度 T 以及粒子群参数；

步骤 2：随机产生大小为 N 的种群；

步骤 3：根据基本粒子群公式更新粒子的速度和位置；

步骤 4：对步骤 3 产生的粒子群，以交叉概率 P_c 选择粒子形成子种群，实施如下操作，以产生一个新种群：从子种群中随机地两两选取个体 $\boldsymbol{X}_i$ 和 $\boldsymbol{X}_k$，按式（3.28）、式（3.29）进行交叉操作，产生两个新个体 $\boldsymbol{X}_j'$ 和 $\boldsymbol{X}_k'$，计算适应度函数值 $f(\boldsymbol{X}_i)$、$f(\boldsymbol{X}_k)$、$f(\boldsymbol{X}_j')$ 和 $f(\boldsymbol{X}_k')$。

若

$$\min\left\{1,\frac{\mathrm{e}^{-f[\boldsymbol{X}_j'-f(\boldsymbol{X}_j)]}}{T}\right\}>\mathrm{rand}()$$

则把 $\boldsymbol{X}_j'$ 作为新个体，若

$$\min\left\{1,\frac{\mathrm{e}^{-f[\boldsymbol{X}_j'-f(\boldsymbol{X}_k)]}}{T}\right\}>\mathrm{rand}()$$

则把 $\boldsymbol{X}_k'$ 作为新个体，其中 rand() 为[0,1]区间上的随机数；

步骤 5：对交叉后产生的新种群按变异概率 P_m 选择粒子形成子种群，实施如下操作以产生一个新种群：从子种群中选取个体 $\boldsymbol{X}_j$，按式（3.32）进行高斯变异操作，产生一个新个体 X_j'，计算适应度函数值 $f(\boldsymbol{X}_j)$ 和 $f(\boldsymbol{X}_j')$，若

$$\min\left\{1,\frac{\mathrm{e}^{-f(\boldsymbol{X}_j'-f(\boldsymbol{X}_j)]}}{T}\right\}>\mathrm{rand}()$$

则把 $\boldsymbol{X}_j'$ 作为新个体；

步骤 6：若当前最优个体满足收敛条件，则进化过程成功结束，返回全局最优解；

步骤 7：若进化次数小于预定最大进化次数，则修改种群的退火温度，即令 $T\leftarrow CT$，转步骤（5）。

高鹰等以表 3.13 中的 4 个基准测试函数的最小值为例，通过计算机仿真来评价比较了该算法与标准粒子群算法的性能。算法的初始化参数为：粒子群规模 20，学习因子 $c_1 = c_2 = 1$，交叉概率 $P_c = 0.5$，变异概率 $P_m = 0.05$，温度冷却系数 $C = 0.8$、退火初始温度 $T = 1000000$。

表 3.13　测试函数

函　数	表　达　式
f_1	$f_1(x,y) = x^2 + y^2$，$-5.12 \leqslant x$，$y \leqslant 5.12$
f_2	$f_2(x,y) = 100(x^2 + y^2) + (1-x)^2$，$-2.048 \leqslant x$，$y \leqslant 2.048$
f_3	$f_3(x,y) = x^2 - 0.4\cos(3\pi x) + 2y^2 - 0.6\cos(4\pi y) = 1$，$-10 \leqslant x$，$y \leqslant 10$
f_4	$f_4(x,y) = \frac{1}{4000}(x^2 + y^2) - \cos(x)\cos(y/\sqrt{2}) + 1$，$-600 \leqslant x$，$y \leqslant 600$

为评价算法的收敛性能，迭代次数设为 1000，连续运行 50 次所得函数全局最小值点的平均值和全局最小值的平均值作为算法的衡量指标。图 3.12 至图 3.15 为对应测试函数的迭代收敛曲线（50 次平均值）。图中横轴表示进化次数，纵轴表示适应度值的对数（即每次进化所得全局最小值的对数）。各图中，上方的一条曲线对应于基本 PSO 算法，而下方的一条曲线对应于 SA-PSO 算法。可以看出，基于 SA-PSO 算法的收敛性能明显优于 PSO 算法的收敛性能。表 3.14 是 PSO 算法和 SA-PSO 算法数值仿真结果，可以看出，SA-PSO 算法对 4 个函数的求解结果均优于 PSO 算法的求解结果（50 次独立运行的平均）。对其他函数所做的大量计算机仿真结果亦说明了这一点。

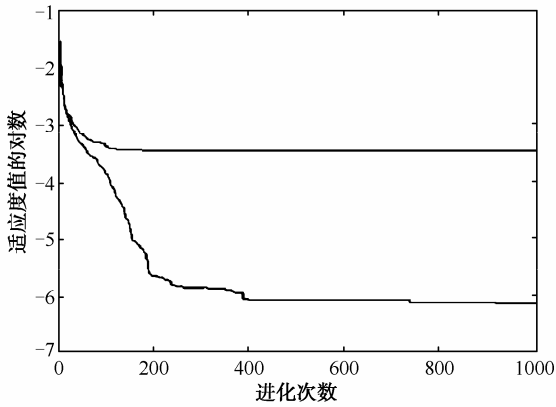

图 3.12　求函数 f_1 最小值的进化曲线

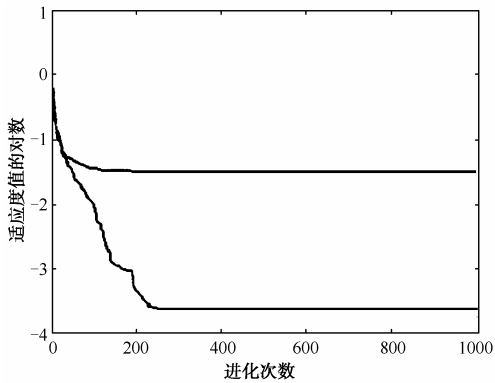

图 3.13　求函数 f_2 最小值的进化曲线

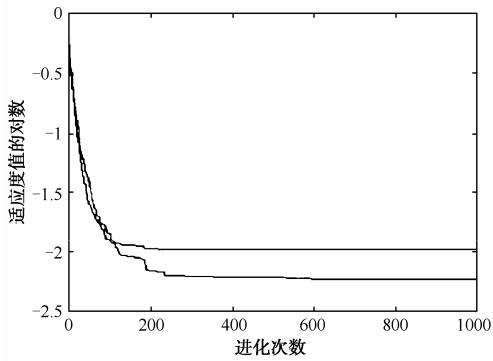

图 3.13　求函数 f_3 最小值的进化曲线

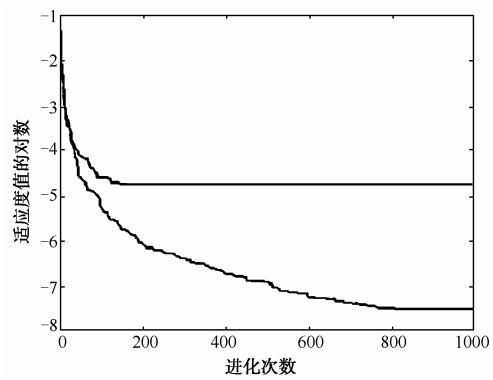

图 3.14　求函数 f_4 最小值的进化曲线

表 3.14　基本 PSO 算法和 SA-PSO 算法数值仿真结果

函数	PSO		SA-PSO	
	全局最小值点平均值	全局最小值平均值	全局最小值点平均值	全局最小值平均值
f_1	（−0.00382479，−0.00151192）	0.00003426	（0.00003311，−0.00006472）	0.00000007
f_2	（0.98091619，0.98617958）	0.03015057	（1.00313646，1.00651501）	0.00002428
f_3	（−0.00146766，0.00047434）	0.01021712	（−0.00102071，0.00039764）	0.00293827
f_4	（−0.00045079，−0.00360870）	0.00000822	（0.00001888，0.00000301）	0.00000003

3.5　PSACO算法

P. S. Shelokar[18]等人于 2007 年提出了基于蚁群和粒子群算法的混合算法，即 PSACO（Particle Swarm Ant Colony Optimization）算法，用于函数的优化。PSACO 算法由 PSO 算法和蚁群优化（ACO）算法两部分组成，PSO 算法中的粒子等同于 ACO 算法中的蚂蚁。首先根据基本 PSO 算法更新粒子的速度和位置；接下来 ACO 算法用于局部搜索，蚁群利用信息素诱导机制改进粒子的位置向量。在第 t 次迭代步，蚂蚁 i 在粒子群的全局最优解 $\boldsymbol{P}_g$ 附近生成一个新位置 $\boldsymbol{Z}_i(t)$：

$$\boldsymbol{Z}_i(t) = N(\boldsymbol{p}_g, \delta) \tag{3.33}$$

利用式（3.33），可以产生满足均值为 $\boldsymbol{P}_g$，方差为δ的高斯分布的位置向量 $\boldsymbol{Z}_i(t)$。初始化时δ=1（$t=1$时），每次迭代时，令 $\delta = \delta \times d$（$d \in (0.25, 0.997)$）。如果 $\delta < \delta_{\min}$（$0.0001 < \delta_{\min} < 0.01$），则 $\delta = \delta_{\min}$。计算适应度值 $f(\boldsymbol{Z}_i(t))$，如果 $f(\boldsymbol{Z}_i(t)) < f(\boldsymbol{X}_i(t))$，则用 $\boldsymbol{Z}_i(t)$ 取代 $\boldsymbol{X}_i(t)$。迭代的开始时，由于δ较大，蚂蚁可以在全局最优附近较大空间范围内进行搜索，提高了全局搜索能力。因此，PSACO 算法不仅有较快的搜索速度，容易快速到达问题的可行解空间，而且有可能找到全局最优解。算法的具体流程图如下所示。

开始

步骤 1：初始化。初始化粒子群和蚁群算法的参数，包括位置矢量和速度矢量，最大迭代数等；

步骤 2：根据适应度函数计算粒子（蚂蚁）适应度值，评价粒子。初始化粒子群（蚁群）的个体最优和全局最优；

步骤 3：根据基本粒子群公式，更新粒子（蚂蚁）的速度和位置矢量；

步骤 4：计算粒子（蚂蚁）的适应度值；

步骤 5: 利用式(3.33)产生新位置 $\boldsymbol{Z}_i(t)$，如果 $f(\boldsymbol{Z}_i(t)) < f(\boldsymbol{X}_i(t))$，则用 $\boldsymbol{Z}_i(t)$ 取代 $\boldsymbol{X}_i(t)$；

步骤 6: 根据适应度值更新粒子（蚂蚁）的个体最优 $\boldsymbol{P}_i(t)$ 和全局最优 $\boldsymbol{P}_g$;

步骤 7: 终止准则是否满足？是：输出全局最优解 $\boldsymbol{P}_g$；否：返回步骤 3。

为验证 PSACO 算法的性能，用 6 个基准测试集的多峰连续函数进行了测试，并和基本 PSO 算法进行了对比，见表 3.15 和表 3.16。

3.15　算法测试的结果

函数	函数测试结果均值			
	PSACO	CPSO	PSO	GA
f_1	3.0000	3.0000	4.6202	3.147
f_2	0.3979	0.3979	0.4960	0.4021
f_3	3.8628	3.8610	3.8572	3.8571
f_4	3.3198	3.1953	2.8943	3.0212
f_5	1.9999	1.9940	1.9702	1.9645
f_6	186.7309	186.7274	180.3265	182.1840

表 3.16　结果比较

函数	PSACO		CPSO		PSO		GA	
	SR	AVEN	SR	AVEN	SR	AVEN	SR	AVEN
f_1	100	157	100	192	98	1397	98	536
f_2	100	156	100	154	94	743	92	1682
f_3	100	159	90	119	96	183	16	112
f_4	98	263	96	2551	26	3796	94	5727
f_5	100	112	98	653	100	1160	84	238
f_6	100	307	100	360	98	1337	98	1516
Overall	99	192	97	672	85	1436	80	1635

3.6　CPSO算法

混沌（Chaos）是自然界广泛存在的一种非线性现象，混沌运动貌似随机，却隐含

着精致的内在结构，具有遍历性、随机性和对初始条件的敏感性等特性，能在一定范围内按其自身规律不重复地遍历所有状态，利用混沌运动的这些性质可以进行优化搜索。

高鹰等[19]把混沌优化思想引入到PSO算法中，提出了混沌粒子群优化算法（PSO算法）。主要措施是利用混沌运动的遍历性以当前整个粒子群迄今为止搜索到的最优位置为基础产生混沌序列，把产生的混沌序列中的最优位置粒子随机替代当前粒子群中的一个粒子的位置。改善了PSO算法易陷入局部极值点，进化后期收敛速度慢，精度较差等的缺点。CPSO算法的具体步骤如下。

开始

步骤1：初始化粒子群。包括算法的参数，迭代次数，混沌寻优次数等；

步骤2：根据基本粒子群算法公式更新粒子的速度和位置；

步骤3：对全局最优解 $\boldsymbol{P}_g$ 进行混沌优化。将 $\boldsymbol{P}_g$ 中的每个元素 p_{gi}（$i=1,2,\mathrm{L}\ ,n$）映射到Logistic（）方程的定义域[0，1]；$z_i=\dfrac{p_{gi}-a_i}{b_i-a_i}$，$(i=1,2,\mathrm{L}\ ,n)$，然后，用Logistic（）方程进行迭代产生混沌变量序列 $z_i(m)(m=1,2,\mathrm{L}\ )$，再把产生的混沌变量序列通过逆映射 $p_{gi}=a_i+(b_i-a_i)z_i(m)$ 回到原解空间，得 $\boldsymbol{P}_g(m)$。在原解空间对混沌变量经历的每一个可行解 $P_g^{(m)}$ 计算适应度值，保留适应度值最好的可行解 P^*；

步骤4：随机从当前群体中选出的一个粒子用 P^* 取代；

步骤5：若达到最大代数或得到满意解，则优化过程结束，否则返回步骤2；

结束

以求一个基准测试函数的最小值为例，通过计算机仿真来评价比CPSO算法和PSO 算法的性能，并和带惯性因子的粒子群优化算法（IWPSO）、杂交粒子群优化算法（Crossover PSO，CRPSO）和带高斯变异的粒子群优化算法（MPSO）进行比较，基准测试函数为：

$$f_3(x,y)=x^2-0.4\cos(3\pi x)+2y^2-0.6\cos(4\pi y)-1$$

其中，$-10\leqslant x$，$y\leqslant 10$，在[−10，10]区间内有1个全局最小值点（0，0）。

算法的初始化参数为：粒子群规模20，学习因子 $c_1=c_2=1$；IWPSO算法中的惯性因子 $w=0.9$；CRPSO算法中的交叉概率 $P_c=0.5$，MPSO化算法中的变异概率

$P_m = 0.05$。为评价算法的收敛性能，进化次数设为 1000，混沌寻优次数为 50。连续运行 50 次所得函数全局最小值点的平均值和全局最小值的平均值作为算法的衡量指标。

图 3.17 是函数最优适应度值的对数（即每次进化所得全局最小值的对数）随进化次数变化的曲线的对比图（50 次独立运行的平均），图中上方做一条曲线对应于 PSO 算法，而下方的一条曲线的对应 CPSO 算法。可以看出，CPSO 算法的收敛性能明显优于 PSO 算法的收敛性能。表 3.17 是 PSO 算法和 CPSO 算法数值仿真结果，可以看出，CPSO 算法对函数的求解结果优于 PSO 算法的求解结果（50 次独立运行的平均）。

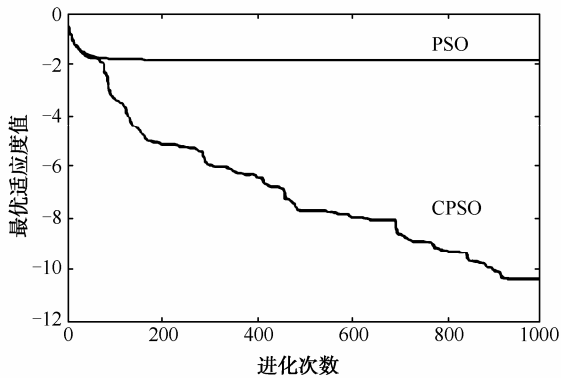

图 3.17　PSO 和 CPSO 算法比较（最优适应度值随进化次数变化的曲线）

表 3.19　PSO 和 CPSO 算法数值仿真结果

算　　法	全局最小值点平均值	全局最小值平均值
PSO	(0.01234485，0.00233262)	0.01575119
CPSO	(1464×10^{-7}，0.623×10^{-7})	4.3978×10^{-11}

参 考 文 献

[1] Kennedy J, Eberhart R.C. A Discrete Binary Version of the Particle Swarm Algorithm.In: The 1997 Conference on System, Cybernetics and Informatics. Piscataway, NJ: IEEE Press, 1997: 4104～4109.

[2] Clerc M. Discrete Particle Swarm Optimization, Illustrated by the Traveling

Salesman Problem. http://www. mau-ricecierc.net, 2000.

[3] Afshinmanesh F, Marandi A, Rahimi-Kian A. A Novel Binar Particle Swarm Optimization Method Using Artificial Immun System. UROCON, 2005.

[4] 周雅兰，王甲海，印鉴．一种基于分布估计的离散粒子群优化算法．电子学报，Vol.36 No.6 June（2008），1242～1248.

[5] 庞巍，王康平，周春光，黄岚，季晓辉．模糊离散粒子群优化算法求解旅行商问题．小型微型计算机系统，第 26 卷第 8 期，2005（8），1331～1334

[6] 叶东毅，廖建坤．最小约简问题的一个免疫离散粒子群算法．小型微型计算机系统，第 29 卷 6 期，2008（6），1089～1092.

[7] Sun J, Feng B, Xu W. Particle Swarm Optimization with Particle Having Quantum Behavior, IEEE Proc. of Congress on Evolutionar Computation, 2004: 325-331

[8] Yang S, Wang M, Jiao L. A quantum particle swarm optimization. Proceeding of the IEEE Congress on Evolutionary Computation, 2004（1），320～324.

[9] Chen A L, Yang G K, Wu Z M. Hybrid discrete particle swarm optimization algorithm for capacited vehicle routing problem. 浙江大学学报，A 卷英文版，2006，7（4），607～614.

[10] Brits，R.，A. P. Engelbreeht，F. Vanden Bergh. A Niching Partiele Swarm Optimiler. In Proeeedings of the Conference on Simulated Evolution and Learning. November 2002.

[11] Evanden Bearch. Ananalysis of Particle Swarm OptimiZaters.University of Pretoria, 2001.

[12] R. Brits，A. P. Engelbrecht, F. Vanden Bergh. A Niching Particle Swarm Optimization. Proc Conf. on Simulated Evolution and Leaming .Singapore，2002，692～696.

[13] 王俊年，申群太，沈洪远，周鲜成．一种基于聚类的小生境微粒群算法．信息与控制，第 134 卷 34 期，680～689.

[14] 贾东立，张家树．基于混沌变异的小生境粒子群算法． 控制与决策，第 22 卷第 1 期，2007（1），117～120

[15] S. Dasctal.. Particle Swarm Optimization and Differential Evolution Algorithms: Technical Analysis, Applications and Hybridization Prespectives, Studies in Computional Intelligence, .2008.

[16] 高鹰，谢胜利. 基于模拟退火的粒子群优化算法. 计算机工程与应用，2004（1），47～50.

[17] P. S. Shelokar, Patrick Siarry , V. K. Jayaraman , B. D. Kulkarni. Particle swarm and ant colony algorithms hybridized for improved continuous optimization. Applied Mathematics and Computation，188 (2007)，129～142.

[18] 高鹰，谢胜利. 混沌粒子群优化算法. 计算机科学，第 31 卷第 8 期，2004，13～15.

第 4 章　PSO算法在机械工程领域的应用

机械工业是为国民经济提供装备的基础工业，任何现代的产业和人们的日常生活都离不开机械工业，在机械工程领域存在着许多的优化问题。随着科学技术的发展，机械工程领域的这些优化问题变得越来越复杂，为了解决这些问题，许多人工智能技术和方法被应用到机械工程领域。本章主要介绍粒子群优化算法在机械工程领域的应用。

4.1　PSO算法在机械故障诊断方面的应用

在现代化生产中，机械设备的故障诊断技术越来越受到重视，如果某台设备出现故障而又未能及时发现和排除，其结果不仅会导致设备本身损坏，甚至可能造成机毁人亡的严重后果。在连续生产系统中，如果某台关键设备因故障而不能运行，往往会影响全厂生产系统设备的运行，而造成巨大的经济损失。因此，对于连续生产系统，故障诊断具有极为重要的意义[1]。人工神经网络（Artificial Neuron Networks，ANN）模型是在现代神经生理学和心理学的研究基础上，模仿人的大脑神经元结构特性而建立的一种非线性动力学网络。作为一种新的模式识别技术或一种知识处理方法，人工神经网络在故障诊断领域中显示了其极大的应用潜力，被广泛应用到各种机械的故障诊断中。

本节针对神经网络在故障诊断中存在的问题，介绍 PSO 算法和神经网络结合的改进方法在机械故障诊断中的应用。

4.1.1　人工神经网络

人工神经网络是高度复杂的非线性动力学系统，它是由大量简单的处理单元——神经元广泛地相互连接而形成的复杂的网络结构。它反映了人脑功能许许多多的基本特性，但它不是人脑神经网络系统的实际再现，而仅仅是对人脑某些机能的抽象、简化及模拟。这也是现实中所能够做到的。研究人工神经网络的目的是探索人脑加工、储存和处理信息的机制，从而开发出具有人脑智能的仿真机器，以实现采用一

般方法难以实现的功能。

虽然组成神经网络的每一个神经元的结构和功能比较简单，但是大量网络构成的系统的行为却丰富多彩，十分复杂。人工神经网络具有一般非线性系统的共性，但其个性特征也十分显著，例如，神经网络具有高维性，神经元之间具有广泛的连续性、自适应性及自组织性等[2]。此外，神经网络还具有很强的容错性和鲁棒性，善于联想、综合和推广，因此在智能控制、模式识别、自适应滤波、信息处理等领域得到广泛应用。

4.1.2　BP神经网络

在网络训练中，对权值进行调整和修正的方法是误差反向传播算法（Back-Propagation，BP）。BP 算法是 Rumelhart 等人在 1986 年提出来的，由于结构简单，可调节的参数多，训练算法多，可操作性好，在实际中得到了广泛的应用。BP 神经网络在结构上类似于多层感知器，是一种多层前馈神经网络。

BP 神经网络的结构如图 4.1 所示，是一种具有三层或三层以上神经元结构的网络，包括输入层、若干个隐含层和输出层。各层之间是一种全连接的形式，但同一层的神经元之间没有连接关系。当一对训练样本用 BP 神经网络进行训练时，神经元的激活值从输入层经中间层向输出层传播，在输出层获得网络已输入的响应。接下来，根据实际输出值和目标输出值之间的误差，从输出层反向进行修正，依次调节中间层和输出层之间的连接权值、阈值、输入层和中间层之间的连接权值和阈值，使误差逐渐减小，这样就从输出层又回到了输入层。这种算法称为误差反向传播算法，即 BP 算法。随着这种误差反向修正的不断进行，网络对输入模式响应的正确率不断得到提高[3]。

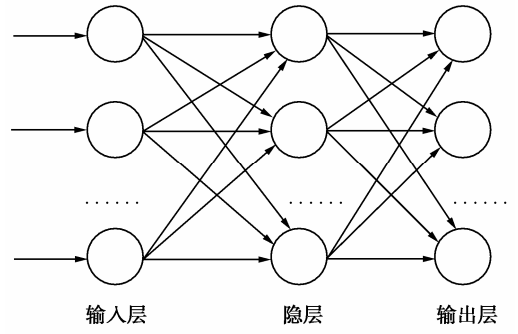

图 4.1　BP 神经网络结构

与感知器不同，由于误差反向传播中会对传递函数进行求导计算，BP 神经网络的传递函数要求必须是可微的，所以不能使感知器网络中的硬阈值传递函数。常用的有 Sigmoid 型的对数、正切函数或线性函数。由于传递函数是处处可微的，所以对于 BP 神经网络来说，一方面，所划分的区域不再是一个线性划分，而是由一个非线性超平面组成的区域，它是比较平滑的曲面，因此它的分类比线性划分更加精确，容错性也比线性划分更好；另一方面，网络可以严格采用梯度下降进行学习，权值修正的解析式十分明确。

三层 BP 神经网络的学习过程与步骤如下。

开始

步骤 1: 在区间[-1，1]上随机初始化连接权值和阈值 w_{ii}、v_{it}、θ_i 和 γ_t；

步骤 2: 随机抽取一组输入和目标样本提供给网络；

步骤 3: 计算中间层的输入向量和输出向量：

$$s_j = \sum_{i=1}^{n} w_{ij} x_i - \theta_j$$

$$b_j = f(s_j)$$

步骤 4: 计算输出层的输入量和输出量：

$$l_t = \sum_{i=1}^{n} v_{jt} b_j - \gamma_t$$

$$y_t = f(l_t)$$

步骤 5: 计算输出层各单元的误差：

$$d_t^k = (y_t^k - C_t) \cdot (1 - C_t)$$

步骤 6: 计算中间层各单元的误差：

$$e_j^k = [\sum_{t=1}^{q} d_t \cdot v_{jt}] b_j (1 - b_j)$$

步骤 7: 修正连接权值 v_{it} 和阈值 γ_t：

$$v_{jt}(N+1) = v_{jt}(N) + \alpha \cdot d_t^k \cdot b_j$$

$$\gamma_t(N+1) = \gamma_t(N) + \alpha \cdot d_t^k$$

$$0 < \alpha < 1$$

步骤 8：修正连接权值 w_{ij} 和阈值 θ_j：

$$w_{ij}(N+1)=w_{ij}(N)+\beta\cdot e_j^k\cdot a_i^k$$

$$\theta_j(N+1)=\theta_j(N)+\beta\cdot e_t^k$$

$$0<\beta<1$$

步骤 9：随机取下一组学习样本进行训练，返回步骤 3，直到所有样本训练完毕；

步骤 10：迭代次数或训练误差是否满足要求？否：返回步骤 2，是：结束训练。

结束

在训练结束后，用测试样本来测试网络的性能。测试样本中应包括实际应用过程可能遇到的所有模式。为了保证网络的泛化能力，在测试样本中不应包含用于训练的学习样本。实际应用过程中，由于一个三层的 BP 神经网络可以完成任意的 n 维到 m 维的映射，所以大部分的 BP 神经网络都选用三层，包括输入层、隐含层和输出层。对于输入层和输出层来说，可以根据具体的问题来确定，而隐含层的确定则比较复杂。隐含层节点的数目与问题的要求，输入/输出节点的数目都有直接关系。隐含层节点数目太多会导致训练时间过长，而且误差不一定最佳，也会导致容错性差，不能识别以前没有见过的样本。对于隐含层节点数目的确定有一些公式可以参考。

（1）$\sum_{i=0}^{n}C_{n_1}^{i}>k$，$k$ 为样本数，n_1 为隐含层节点数，n 为输入单元数。如果 $i>n_1$，$C_{n_1}^{i}=0$。

（2）$n_1=\sqrt{n+m}+a$，m 为输出神经元数，n 为输入神经元数，a 为[1，10]区间的常数。

（3）$n_1=\log_2 n$，n 为输入单元数。

除上述公式外，还可以采用动态调节的方法来确定隐含层的单元数，例如，在开始放入足够的隐含层单元，通过学习训练剔除那些不起作用的单元；或者通过优化算法，例如 PSO 算法来动态确定隐含层单元数目。

BP 神经网络由于在多个参数需要调节，因此其自身存在着一些缺陷，主要包括以下几个方面。

（1）由于学习速率是固定的，因此，网络收敛速度比较慢，需要较长的训练时间。目前通常采用附加动量项、变化的学习速率或自适应的学习速率的方法得以改善。

（2）容易陷入到局部最小，不能保证收敛到某个指定的值。这是因为采用梯度下降法可能会产生多个局部最小值。

（3）网络的学习和记忆具有不稳定性。也就是说，如果增加学习样本，训练好的网络就需要从头开始重新训练，对以前的权值和阈值没有记忆功能。

（4）网络隐含层的层数和神经元数目的确定尚无理论上的指导，需要根据经验或者通过反复实验来确定。

4.1.3　人工神经网络与PSO算法

人工神经网络是模拟大脑分析过程的简单数学模型，BP 算法是最流行的神经网络训练算法。如前所述，由于 BP 神经网络存在诸多的缺陷，为了改善算法的性能，近年来也有很多研究开始利用进化计算技术来研究 BP 算法和其他神经网络的各个方面。进化计算主要用来研究神经网络的三个方面：网络连接权重，网络结构（网络拓扑结构、传递函数），网络学习算法。

不过大多数这方面的工作都集中在网络连接权重和网络拓扑结构上。在 GA 算法中，网络权重和（或）拓扑结构一般编码为染色体（Chromosome），适应度函数（Fitness Function）的选择一般根据研究目的确定。例如在分类问题中，错误分类的比率可以用来作为适应度值。

进化计算的优势在于可以处理一些传统方法不能处理的例子，例如不可导的节点传递函数或者没有梯度信息存在的问题。但是缺点在于：① 在某些问题上性能并不是特别好；② 网络权重的编码以及遗传算子的选择有时比较麻烦。

近年来研究人员开始用 PSO 算法来代替 BP 算法来训练神经网络，研究表明 PSO 算法是一种很有潜力的神经网络训练算法。PSO 算法的速度比较快而且和传统的 BP 算法相比更容易收敛到全局最优，同时也不会出现 GA 算法所碰到的问题。下面简单介绍如何利用 PSO 算法训练神经网络。

对于神经网络来说，主要的工作是动态调整网络的权重，使得输出的均方误差能量最小。对于 PSO 算法来说，主要的工作过程是不断更新粒子的速度和位置，更新的依据是适应度函数的计算结果（粒子的适应度值）。因此，可以利用神经网络的权重和阈值来构成粒子的位置矢量，利用均方误差能量函数作为粒子群的适应度函数[4]，即粒子群中第 k 个粒子的维数为

$$\boldsymbol{X}_k = [\, w_{ij}\,,\ v_{jt}\,,\ \theta_j\,,\ \gamma_t\,] \tag{4.1}$$

其中，$i=1,2,\cdots,I$，$j=1,2,\cdots,J$，$t=1,2,\cdots,T$；I、J 和 T 分别为输入层、隐含层和输出层节点数；$w_{j,i}$、$v_{j,t}$ 为输入层、中间层、隐含层之间的连接权值；θ_j、γ_t 为隐含层和输出层的阈值。

以各层之间的连接权值和阈值确定的网络输出与期望值的方差作为种群的适应度函数，即

$$J(t)=\frac{1}{N}\sum_{j=1}^{N}(y_j(t)-\hat{y}_j(t)) \tag{4.2}$$

其中，N 为训练样本数；$\hat{y}_j(t)$ 为第 t 次迭代第 j 个训练样本输入的网络实际输出；$y_j(t)$ 为期望输出值。

假设有一个分类问题：输入变量为 4 个，输出变量为 3 个。利用三层 BP 神经网络进行分类时，可以知道，这个神经网络的输入层有 4 个节点，对应了 4 个输入变量；输出层有 3 个节点，对应了 3 个输出变量。对于隐含层的节点数目，我们可以动态调节，不过为了说明问题，这里假定隐含层有 6 个节点。如果用 PSO 算法对神经网络进行训练，则粒子的维数为 $4\times6+6\times3+6+3=51$。PSO 训练神经网络的步骤如下[4]。

开始

步骤 1：根据问题确定神经网络结构；
步骤 2：根据式（4.1）确定粒子的维数；
步骤 3：根据式（4.2）确定粒子群的适应度函数；
步骤 4：随机初始化粒子群；
步骤 5：用 PSO 算法训练神经网络，直到满足停止准则；
步骤 6：输出最好的粒子。

结束

4.1.4　应用案例

传统的 BP 算法由于存在着容易陷入局部最优的缺点，因此，许多研究都将 PSO 算法用来训练 BP 神经网络，并应用于各种机械的故障诊断中[4,5,6,7,8]。

旋转机械广泛应用于各种机械系统中，当其发生故障时会降低系统的工作性能，甚至导致系统崩溃。齿轮箱是旋转机械中应用最为广泛也是最易损坏的机械部件。旋转机械的许多故障都与齿轮箱有关。齿轮箱发生故障缺陷时会导致设备产生异常

振动和噪声，甚至可能造成整台设备的损坏。因此，可以通过测量齿轮箱上的振动信号来分析设备的运行状况。

在机械运转过程中，由于齿轮箱本身的结构特点、加工装配误差及运行过程中出现的故障等内部因素，以及传动轴上其他零部件的运动和力的作用等外部因素，当齿轮箱以一定的速度并在一定的载荷下运转时，对齿轮箱系统产生激励，使该系统振动。在实际诊断中，如果不考虑齿轮箱的加工和装配误差，则振动主要是运行故障这一内部因素所引起的。

振动法是通过安装在齿轮箱轴承座或者箱体适当部位的振动传感器监测振动信号，并对此信号进行分析与处理来判断齿轮箱状态。振动法具有适用于各种类型工况的齿轮箱，可以有效地诊断出早期微小故障，信号测试与处理简单、直观，诊断结果可靠等优点，所以在实际中得到了极为广泛的应用。振动信号是齿轮箱故障的载体，理论上可通过齿轮箱产生的振动信号的分析和处理诊断出所有类型的故障。但实际上，通过传感器所拾取的振动信号除反映有关齿轮箱本身的工作情况外，还包含了大量其他运行部件和结构的噪声等振动信息。因此，突出故障特征信息，抑制背景噪声，从而有效诊断出齿轮箱故障成为齿轮箱振动监测与诊断技术的重要部分。齿轮箱故障诊断的基本步骤一般包括：采集齿轮箱的振动信号，并将振动信号处理成时域或频域特征参数；采用合适的故障诊断方法，根据齿轮箱的时域和频域特征参数判断齿轮箱是否存在缺陷。

为了获得齿轮箱振动信号的实验数据，相应的测试工作在中北大学振动实验室的实验台架上完成，实验以 JZQ250 型齿轮箱为研究对象，该齿轮箱为二级传动齿轮减速器，总传动比为 10.35。齿轮箱主要由输入轴、中间轴、输出轴、两对斜齿轮、三对滚动轴承和箱体所组成。输入轴经由联轴器与电机相连，轴两端支撑在滚动轴承上；输入轴通过一对斜齿轮把动力传递到中间轴，中间轴由一对滚动轴承、两个斜齿轮和轴组成；输出轴与负载轮键相连接。齿轮箱系统简图如图 4.2 所示。实验系统由 Y132S-4 三相异步电动机驱动，电机输出轴经由联轴器与齿轮箱的输入轴相连，再由齿轮箱传动，传递动力于负载轮，最后经由抱紧装置锁死。本实验测试系统包括：加速度振动信号测试系统，转矩、转速测试系统。其测试系统的框图如图 4.3 所示。

根据经验和实际测试，选定齿轮减速器轴承座附近作为加速度传感器的安装位置。因此，测试过程中，加速度传感器被固定在轴承座上用来测量齿轮工作时的振动信号。

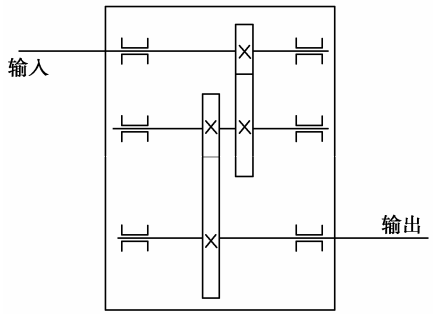

图 4.2　齿轮箱系统简图

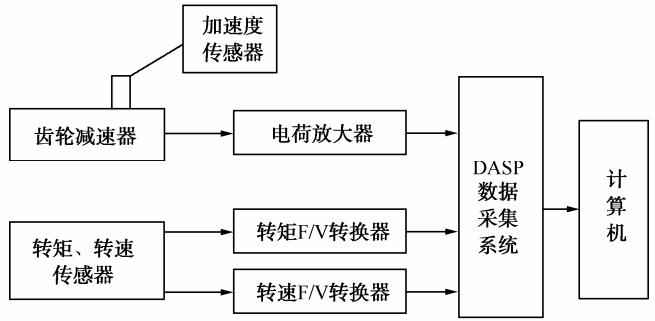

图 4.3　测试系统框图

在对测量获得的振动信号进行预处理后，选取了时域和频域指标作为特征量。时域特征参数包括脉冲、均方根、峭度、裕度 4 个参数；频域特征参数选取齿轮箱中间轴齿轮的旋转频率及其 2、3 次谐波处的功率谱分量（pfr1，pfr2，pfr3），啮合频率及其 2、3 次谐波处的功率谱分量（pfg1，pfg2，pfg3），啮合频率的边频带 fg±nfr（n=1，2）处的功率谱分量（pf1，pf2，pf3），共 10 个。

用 PSO-DV 算法优化神经网络，首先要建立合理的神经网络模型和粒子的模型，确定粒子的适应度函数和搜索空间范围。选择三层神经网络，输入层的节点数目为 14 个，分别对应了 4 个时域特征参数和 10 个频率段的功率谱；输出层的节点数目为 3 个，分别对应了齿轮的 3 种工况，即正常、齿面点蚀、崩齿，分别编码为 001、010 和 100。根算法和试验，确定网络的隐含层神经元数目为 13 个，因此，神经网络的结构为 12-13-3。

在神经网络的结构确定后，就可以确定种群中各粒子的维数。根据式（4.1）种

群中第 k 个粒子 $\boldsymbol{X}_k$ 的维数为 $14\times13+13\times3+13+3=237$ 。同时，以各个层之间的连接权值和阈值确定的网络输出与期望值的方差作为种群的适应度函数。

为了说明标准 PSO 算法的缺点和一些改进算法的优点，在实验过程中分别用标准 PSO 优化的 BP 算法和 PSO-DV 算法优化的 BP 神经网络对样本数据进行训练，达到要求的收敛精度或者达到最大迭代次数，结束神经网络训练。选取 60 组样本数据进行归一化处理、编码，然后用不同于样本的 20 组故障样本进行测试。

每种算法在相同的初始化条件下连续运行 10 次，记录每次运行时的收敛误差（MSE）和连续运行结束后，两种算法粒子群陷入局部极小的次数，记录在表 4.1 中。图 4.4 所示为标准 PSO 算法和 PSO-DV 算法在相同初始化条件下优化的 BP 网络的训练误差曲线（仅给出标准 PSO 算法陷入局部最优的对比曲线）。训练过程中所用的参数、训练的结果见表 4.2。测试样本用两种算法训练的神经网络进行测试，结果见表 4.3。

表 4.1　标准 PSO 算法和 PSO-DV 算法的训练参数和性能比较

算法	标准 PSO	PSO-DV
训练参数	swarm size: 45	swarm size: 45
	c_1=c_2=2.0	β=0.5
	w=0.7	C_R=0.1～0.9,依迭代时间线性增长
	Max iteration: 2000	Max iteration: 2000
	SSM: 0.001	SSM: 0.001
诊断精度(%)	86.7	93.3
陷入局部最优次数	6	0

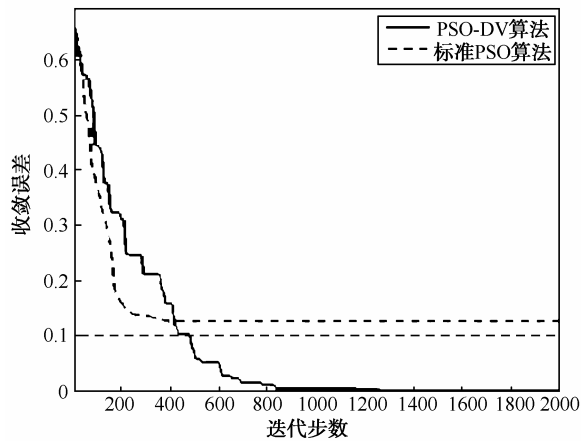

图 4.4　PSO 算法和 PSO-DV 算法训练 BP 网络收敛曲线比较

表 4.2　测试样本集

参　数	样 本 编 号					
	1	2	3	4	5	6
峭度	0.19652	0.60136	0.38977	0.69851	0.72365	0.78452
裕度	0.36338	0.28576	0.27895	0.20156	0.62939	0.58263
脉冲	0.49366	0.47512	0.54658	0.40221	0.54366	0.42783
均方根	0.29856	0.18788	0.25055	0.55183	0.37484	0.44308
pfg1	0.42551	0.32658	0.49779	0.56214	0.30906	0.29572
pfg2	0.25003	0.59906	0.82149	0.30508	0.40215	0.62988
pfg3	0.80277	0.58042	0.89558	0.53025	0.57541	0.54081
pf1	0.34564	0.75238	0.30584	0.58869	0.20462	0.19855
pf2	0.30225	0.58966	0.60574	0.81251	0.72508	0.88450
pf3	0.49664	0.51218	0.50284	0.41502	0.63257	0.44504
pf4	0.54377	0.49992	0.53076	0.15452	0.68702	0.29547
pfr1	0.80036	0.45221	0.14898	0.70169	0.25184	0.76588
pfr2	0.29844	0.35895	0.67229	0.50323	0.61575	0.10441
pfr3	0.58542	0.11226	0.80442	0.62584	0.25703	0.48552

表 4.3　测试样本输出结果比较

样本编号	故障类型	PSO 算法输出	PSO-DV 算法输出	输出圆整
1	正常	(0.98932, 0.00005, 0.00253)	(0.99285, 0.00213, 0.12032)	(1,0,0)
2	正常	(0.95604, 0.00069, 0.00170)	(0.97128, 0.00154, 0.01683)	(1,0,0)
3	点蚀	(0.00182, 0.98135, 0.02302)	(0.00091, 0.99632, 0.00110)	(0,1,0)
4	点蚀	(0.00814, 0.99360, 0.00259)	(0.00305, 0.98370, 0.00570)	(0,1,0)
5	崩齿	(0.00528, 0.00012, 0.98561)	(0.01043, 0.00250, 0.99824)	(0,0,1)
6	崩齿	(0.00447, 0.00035, 0.96384)	(0.00328, 0.00159, 0.98896)	(0,0,1)

从表 4.1 可看出，标准 PSO 算法确实比较容易陷入局部最优，共有 6 次陷入了局部最优，而改进的 PSO-DV 算法没有一次陷入局部最优。从图 4.4 可以看出，标准 PSO 算法优化的神经网络在达到一定迭代次数后，仍然没有收敛到要求的误差，且从某次迭代后开始，误差基本不再发生变化，陷入了局部极小；而改进的 PSO-DV 算法最终都收敛了设定的收敛误差，避免了训练陷入局部极小，算法的收敛性能得

到了极大的改善。同时从图 4.4 还可以发现，利用标准 PSO 算法训练的 BP 神经网络的收敛曲线都比较光滑，而改进的 PSO-DV 算法的收敛曲线有“抖动”的情况，这也充分说明 PSO-DV 算法能够接受更坏的情况，跳出局部极小。之所以会出现这种情况，是由于 PSO-DV 算法引入了 DE 算法，“自然选择”的策略增加了种群的潜在多样性。

表 4.2 的统计结果表明，PSO-DV 算法和标准 PSO 算法一样，不仅能很好地辨识出各类故障，而且网络的实际输出值与网络期望值非常接近，说明 PSO-DV 算法对故障信号具有较强的识别能力，同时，陷入局部极小的概率相比标准的 PSO 算法大大降低。

实验结果表明，PSO-DV 算法训练 BP 神经网络，克服了 PSO 算法在到达目标局部极小值附近时，可能出现收敛速度慢，在极小点附近振荡，甚至会陷入局部极小的缺点，收剑效果上得到了显著的提高。当然，在训练的早期，PSO-DV 方法所用的时间比 PSO 确实要多一些。

4.2　PSO算法在机械测试中的应用

长期以来，人们对机械设备实施基于振动信号的故障检测时，由于受到理论和技术的限制，检测振动信号的传感器不能有效地采集故障征兆，难于优化配置传感检测系统，故障的严重程度和故障的发生部位不能准确确定。另一方面，在对车辆或装备的齿轮传动箱进行性能监测和质量监控时，故障的出现往往具有关联性，需要确定多部位、不同零部件和不同类型的故障，从而使有效地检测和诊断车辆齿轮传动箱的工作状态成为一个客观难题，影响了车辆整个系统的监控质量。考虑到经济和结构运行状态等方面的原因，在整个传动箱上安置尽可能多的传感器来采集故障信号是不可能也是不现实的，因此，就出现了传感器的优化配置问题。通过用尽可能少的传感器（测点）来获取最可靠而最全面的传动箱状况信息，达到最高的故障确诊率，就是测点优化配置的主要目的。

潘宏侠等[9]提出了基于小波神经网络和 PSO 算法的方法来解决测点优化布置问题。该方法直接以多个测点的测试数据作为特征值，以故障诊断的诊断误差为目标函数，根据各测点信号特征之间的相似程度来衡量各测点间的关联程度，建立各测点间相关性的网络模型进行测点优化，从而找出影响传动箱诊断精度的主要因素。

在齿轮箱的故障诊断中，潘宏侠等对这种方法进行了实验验证。按照经验初步选取了实验齿轮箱轴承座附近的 6 个测点，作为测点优化配置的测点布置方案。选取齿轮箱在正常、崩齿和轴承外圈故障 3 种工况，测试了输入轴转速为 900r/min 和

1200r/min 的振动信号，提取了均方根值、峭度指标、裕度指标、脉冲指标、功率谱重心指标和谐波因子 6 个特征参量进行了实验验证。表 4.4 为小波神经网络输出结果，诊断率的结果见表 4.5。

表 4.4　小波神经网络输出结果（网络的训练结果）

测试样本			网络的输出		
			x_1	x_2	x_3
测点 1	900r/min	正常	3.23×10^{-5}	1	0.000794 662
	1200r/min	正常	5.81×10^{-16}	1	7.02×10^{-10}
	900r/min	轴承外圈故障	3.27E−11	0.0115247	0.8904129
	1200r/min	轴承外圈故障	7.36×10^{-5}	1.49×10^{-6}	0.9999774
	900r/min	崩齿	0.9959282	0.00192974	5.25×10^{-8}
	1200r/min	崩齿	0.999 999 9	5.59×10^{-10}	1.27×10^{-5}
测点 2	900r/min	正常	0.5141068	0.6336444	3.60×10^{-9}
	1200r/min	正常	0.004254727	0.9990901	1.08×10^{-10}
	900r/min	轴承外圈故障	5.4810^{-16}	1	5.11×10^{-10}
	1200r/min	轴承外圈故障	0.9829622	0.1048577	1.01×10^{-5}
	900r/min	崩齿	0.9943477	0.003831541	2.97×10^{-7}
	1200r/min	崩齿	0.001523 876	0.9981714	1.01×10^{-9}
测点 3	900r/min	正常	3.2×10^{-5}	4.3×10^{-8}	0.9999999
	1200r/min	正常	0.9999997	1.23×10^{-8}	0.7029566
	900r/min	轴承外圈故障	0.5692037	0.000415001	0.8076008
	1200r/min	轴承外圈故障	0.787618	8.92×10^{-5}	0.9932099
	900r/min	崩齿	0.6223217	0.002238 561	0.1203508
	1200r/min	崩齿	9.27×10^{-10}	3.40×10^{-6}	0.9999381
测点 4	900r/min	正常	0.370812	0.9999422	0.004676297
	1200r/min	正常	2.89×10^{-7}	0.9999992	6.78×10^{-10}
	900r/min	轴承外圈故障	0.6642637	0.03045125	0.08878457
	1200r/min	轴承外圈故障	6.32×10^{-16}	1	8.91×10^{-10}
	900r/min	崩齿	0.6170945	7.40×10^{-5}	0.9874683
	1200r/min	崩齿	0.01488889	5.98×10^{-5}	0.9969465

续表

测试样本			网络的输出		
			x_1	x_2	x_3
测点 5	900r/min	正常	0.1764137	0.9999985	5.49×10^{-7}
	1200r/min	正常	0.002746655	1	4.96E−27
	900r/min	轴承外圈故障	7.47×10^{-8}	7.74×10^{-7}	0.9999971
	1200r/min	轴承外圈故障	0.0167 843	1.80×10^{-8}	0.952 634 4
	900r/min	崩齿	0.4555314	0.06164249	1.55×10^{-8}
	1200r/min	崩齿	0.9993529	0.00767131	8.48×10^{-11}
测点 6	900r/min	正常	1	1.42×10^{-14}	9.47×10^{-6}
	1200r/min	正常	0.0004788	0.9902691	0.000168 968
	900r/min	轴承外圈故障	0.05512785	0.006659533	0.9950697
	1200r/min	轴承外圈故障	2.46×10^{-7}	3.93×10^{-9}	1
	900r/min	崩齿	0.9240733	0.002525921	0.941509 6
	1200r/min	崩齿	0.9993535	0.007685 084	8.46×10^{-11}

表 4.5　诊断率的结果

内　　容	测点 1	测点 2	测点 3	测点 4	测点 5	测点 6
最大输出误差	0.109 571	1	1	1	0.544 469	1
正确诊断次数	6	2	2	2	5	4
诊断率	100%	33.3%	33.3%	33.3%	83.3%	66.7%

注:诊断率大于 80%为正确，允许误差为 20%。

从优化结果可以看出，测点 2、3 和 4 的故障信息较少，在实验中可以不予考虑。测点 1 和 5 的诊断结果较好，测点 6 信号的故障信息分散度较大，其诊断结果不稳定。此外也可看出，输入轴所含信息较多可为首选的测试点，但由于连接装置的影响，数据的分散度大。在中间轴测点的诊断精度与典型故障设置的位置有关。因为输出轴处于传递的末端，故障信息衰减多，在其上面配置的测点对故障不敏感。实验结果表明：基于 PSO 的测试点优化配置是有效的，可以在复杂结构状态检测和故障诊断中起到指导测试点优化配置的作用，能够减少无效测试点的数量和缩减诊断过程，提高实时诊断的质量和系统的诊断精度。

刘波等人用二进制 PSO 算法结合线性方程组设计了太阳热辐射模拟装置，用来测试某些装备和设备的热防护性能[10,11]。

4.3　PSO算法在机械工程其他领域的应用

（1）并联机器人的位置分析是其误差控制与补偿的前提，特别是位置正解，是机器人机构应用的基础。对于并联机器人机构的位置正解问题，通常需要求解一组非线性方程组，这是并联机器人机构运动分析的难点和重点之一。车林仙等[12]提出一种改进的 PSO 算法——分层搜索自适应变异粒子群算法（Hierarchical Search Adaptive Mutation PSO，HSAMPSO），并将其用于对称结构 Stewart 并联机器人机构的位置正解。结果表明，应用 HSAMPSO 算法求对称结构 Stewart 并联机器人机构的位置正解，能够得到机构的全部位置正解和装配构型，且收敛速度较快、精度较高。

（2）管路系统是连接产品主要结构的部分，其重要的程度就相当于人体的血管。美国通用电气公司对以往研制的发动机在使用中出现的空中停车事件进行归纳总结后，发现引起空中停车事件的真正原因中，50%是由于外部管路、导线、传感器等损坏、失效引起的。现代航空发动机的外部附件系统十分复杂，管路需敷设在机匣和机舱之间本已有限的空间中，这就使外部管路系统的复杂程度大大增加。航空发动机传统的布管方法是制造出全尺寸的金属模型样机，依靠设计人员进行手工敷管，布管具有随意性、周期长等问题。柳强等[13]针对航空发动机的内机匣表面的特点，建立了布管模型坐标系，提出了一种新的管路敷设方法。针对传统变长度编码的不足，设计了一种基于栅格的定长度的粒子编码方法，应用改进的 PSO 算法在环境模型中搜索最优路径，最后进行了单条管路与多条管路的布局仿真，结果证明了该方法的可行性和有效性。

（3）在数控加工中，合理地选择切削参数对提高生产率、降低生产成本有着非常重要的意义。目前，大多数工厂在生产中凭经验或是参考手册来选取切削用量，为了避免和尽可能地减少出现异常，一般都选取比较保守的数值。因此，往往达不到切削参数的最优值。就我国数控技术的发展现状而言，切削参数的选择是困扰数控加工的一个重要问题。刘海江等[14]以实际生产加工中的经验公式为基础，考虑加工中机床和刀具的实际约束，运用金属切削理论、数学建模与模型优化算法寻求切削用量的最优值，是当前切削参数选择的一个重要方向，建立了以进给量和切削速度为变量，以最大生产率和最低生产成本为优化目标的多目标优化模型，并引入协调系数将其转化成单目标优化模型，应用 PSO 算法对数学模型进行寻优求解。实践表明，经过优化计算得到的切削参数取值比经验值更能满足优化目标。

（4）设备布局是制造系统设计的重要组成部分，设备布局是否合理对整个制造系统的总体功效具有非常重要的作用。据统计，因设备布局不合理所产生的运行费用占制造系统整体运行费用的 20%~50%，而优良的设备布局可使这一费用减少 10%~30%。优良的设备布局还能加快物料处理的效率，减少在制品的停留时间，提高企业的生产率，使企业对市场需求做出及时的响应，增强企业的市场竞争力。优良的设备布局可以减少工件缓冲区的容量，缩小制造系统所占用的空间，减少制造系统建设的费用。反之，将会导致过量中间存储、物料流动受阻、生产时间延长和生产效率降低。因此，研究制造系统的设备布局具有重要的理论意义和应用价值。曾议等[15]通过建立设备布局问题的图结构描述，使用 ACS 算法搜索优化解，同时给出了改进的 PSO 算法和 ACS 算法求解单向环形布局问题的具体步骤。通过实例计算和比较，表明该算法能有效地节省生产成本，提高生产效率，对解决设备布局这类问题具有很好的实用价值。

（5）液体动压径向滑动轴承通常用于高速和重载的齿轮箱中，是齿轮箱的重要部件。根据流体动压径向滑动轴承的常规设计方法，轴承的宽径比、轴承孔和轴颈的相对间隙等参数的确定都是按经验在一个取值范围内选取的，因此带来了功耗过大、材料浪费、状态不佳、性能不可靠等问题。要设计一个性能优良、成本最低的流体动压轴承，在总体上必定要体现最大化效益、最小化成本这一基本原则，其本质上属于多标优化设计。罗佑新等[16]在利用灰色综合关联度对混合离散变量多目标优化设计问题的全局极值和个体极值进行选取的基础上，利用改进的 PSO 算法，引入动态罚函数，构造新的适应函数，开发了混合离散变量优化的灰色 PSO 算法程序，实现了高维多目标优化问题求解，并利用该方法对齿轮箱动压轴承进行了优化设计。结果表明应用这种方法可获得比现有其他方法更好的结果。该方法对优化设计问题的特性无特殊要求，具有较好的普适性，而且程序运行可靠，全局收敛能力强。

（6）水轮机控制器是保证水电站水轮发电机组稳定运行的重要控制设备。目前，水轮机控制器基本上都是采用 PID 控制规律。因此，如何选择水轮机控制器的最优调节参数，使得水轮机控制系统具有良好的动态品质，是保证水轮发电机组安全稳定运行及提高供电质量的一个重要的研究课题。水轮机控制器 PID 参数优化常用方法有：梯度法、单纯形法、遗传算法等，各有其优点，但也存在明显的缺陷。梯度法要求目标函数连续可导；单纯形法受初值和计算步的影响较大，易收敛于局部最优解；遗传法需要进行复制、交叉与变异操作，进化速慢，易产生早熟收敛，并且其性能对参数有较大依赖性。方红庆[17]用 IDPSO 算法对水轮机控制器 PID 参数进行优化设计。以水轮机转速偏差的加权 ITAE 指标作为改进 PSO 算法的适应度函数。结果表明，经过改进的 PSO 算法优化后的水轮机控制器 PID 控制规律具有优良的过程性能。

参 考 文 献

[1] 钟秉林，黄仁．机械故障诊断学．北京：机械工业出版社，2002.

[2] 葛哲学，孙志坚．神经网络理论与 MATALAB R2007 实现．北京：电子工业出版社，2007.

[3] Simon Haykin 著，叶世伟，史忠植译．神经网络原理．北京：机械工业出版社，2004.

[4] Bo Liu, Hongxia Pan. A Hybrid PSO-DV Based Intelligent Method for Fault Diagnosis of Gear-box. 2009 IEEE International Symposium on Computational Intelligence in Robotics and Automation(CIRA2009).

[5] Liu Bo, Pan Hongxia. Fault Diagnosis of Bearing Using Wavelet Packet Transform and PSO-DV Based neural network. The 6th International Conference on Natural Computation (ICNC'10).

[6] Wei Xiuye, Pan Hongxia and Ma Qingfeng. Application of Particle Swarm Optimization Based Neural Network to Fault Diagnosis. Journal of Vibration, Measurement & Diagnosis, Vol. 26, Feb. 2006, pp. 133～137.

[7] Liu Bo. Fault Diagnosis of Bearing Using PSO with Differential Operator and Neural Network. The 9th International FLINS Conference on Foundations and Applications of Computational Intelligence (FLINS2010).

[8] Qian-jin, Guo, and Hai-bin. A hybrid PSO-GD based intelligent method for machine diagnosis. Digital Signal Processing , Vol.16, 2006, pp. 402～410.

[9] Bo Liu, Hongxia Pan, Xiuling Li. An Expert System for Fault Diagnosis in Diesel Engine Based on Wavelet Packet Analysis and Hybrid PSO-DV Based Neural Network. 2010 International Conference on Intelligent Computing and Cognitive Informatics (ICICCI 2010).

[10] 刘波，潘宏侠．火炮身管热护套防护效率测试研究．兵工学报，2010（8）.

[11] Bo Liu，Hongxia Pan. A sun thermal radiation simulator based discrete PSO.

[12] 车林仙，何兵，易建，陈长忆，罗佑新．对称结构 Stewart 机构位置正解的改进粒子群算法．农业机械学报，第 39 卷第 10 期，158～163.

[13] 柳强，王成恩，任涛，白晓兰．基于粒子群算法的航空发动机管路布局方法．东北大学学报（自然科学版），第 30 卷第 7 期，2009（7），940～955.

[14] 刘海江，黄炜．基于粒子群算法的数控加工切削参数优化．同济大学学报(自然

科学版)，第 36 卷第 6 期，2008（6），803～806.

[15] 曾议，竺长安．基于群智能算法的设备布局离散优化研究．计算机集成制造系统，第 13 卷第 3 期，2007(3)，541～552.

[16] 罗佑新，车晓毅，汪超．流体动压轴承多目标优化设计的改进灰色微粒群算法．润滑与密封，第 33 卷第 4 期，2008 年 4 月，95～105.

[17] 方红庆．一种改进粒子群算法及其在水轮机控制器 PID 参数优化中的应用．南京理工大学学报（自然科学版），第 32 卷第 3 期，2008(6)，274～277.

第 5 章　PSO算法在其他工程领域的应用

工程实际中存在着众多的优化问题，PSO 算法由于其在求解优化问题中的通用性和有效性，正逐步在实际问题中得到应用，广泛用于函数优化、计算机、通信、过程工业、电力系统和化工过程控制等领域，显示出了强大的应用潜力。本章主要介绍 PSO 算法在一些行业领域的应用。

5.1　函数优化

在优化算法的研究过程中，为了评价算法的性能，例如收敛性、收敛速度、鲁棒性、寻优精度等，人们通常采用各种具有不同性能的基准测试函数（Benchmark 函数）来测试和比较各种算法的性能。这些函数从不同的方面来评价优化算法的性能，因此，其包括了从单峰函数到多峰函数，从低维函数到高维函数的众多不同特点的函数，可以从不同的方面对算法的性能进行测试。常见的测试函数主要包括四类：

① 无约束优化测试函数；
② 有约束优化测试函数；
③ 极大极小优化测试函数；
④ 多目标优化测试函数。

在算法性能的测试中，可以根据不同的需要选用不同的测试函数，由于篇幅关系，不再详细介绍，常用的一些测试函数参见附录 A。

5.2　电力自动化领域的应用

制定停电后的“黑启动”方案是实现系统快速恢复的一种有效方法。在制定“黑启动”方案过程中，除了要对各方案进行可行性校验（主要包括自励磁、过电压、电压与频率稳定、静态稳定与低频振荡等），还需要考虑此方案对系统后续恢复的影响，因为初期最优的方案在后续恢复中的效果不一定最好。为完善初期“黑启动”方案评估的指标体系，应该把后续恢复的效率指标纳入到评估框架中。然而，网络

重构效率的量化指标求取是一个尚未很好解决的问题。国内外在电力系统恢复方面进行了大量研究。刘连志等[1]基于不同“黑启动”方案，以考虑发电量和重要负荷停电损失的网络重构效率作为优化目标，采用交叉 PSO 算法和最短路算法相结合实现节点组合优化，并综合考虑机组的启动时间和升负荷特性，求得最优网络重构方案和效率指标。

电力机组组合问题是电力生产与供应中的重要优化问题，合理的机组组合会大大降低电力系统的运行成本，因此一直是一个被广泛探索和研究的热点问题。从数学的角度来讲,机组组合问题是一个大规模非线性混合整数规划问题。韩恺等[2]提出了一种新的惯性权重整定策略，将经典控制理论中的反馈机制和闭环控制系统的概念引入系统形成了一种闭环 PSO（CLPSO）算法来解决电力系统机组组合问题。CLPSO 算法利用反馈信息及时调整粒子的惯性权重，极大地满足了粒子在搜索过程中的动态特性，保证了种群的多样性，提高了算法的全局搜索能力。仿真结果表明了该算法的优越性。

电力系统的无功功率平衡是保证电力系统电压质量的必要条件，无功优化可以充分利用电力系统中的无功电源，改善电压质量、减少网络损耗和提高电压稳定性。常用的控制手段有带负荷调压变压器、可投切电容器和可调压发电机。从本质上讲，无功优化问题是一个离散的、有约束非线性组合优化问题。赵波等[3]结合 PSO 算法和 MAS 技术构造了一种全新的算法：多智能体 PSO 算法（MAPSO 算法）来求解电力系统无功优化问题。首先构造一个格子环境，所有的 Agent 都生存在这个环境中。每一个 Agent 就是 PSO 算法种群中的一个粒子，它们被固定在一个格子中，通过与其邻居的竞争与合作操作和自学习操作，结合 PSO 算法的进化机制，不断地通过 Agent 间的交互和 Agent 与环境间的相互影响，来更新每个 Agent 在解空间的位置，使其能够更快地、更精确地收敛到全局最优解。最后以 IEEE 30 节点系统为试验系统进行了仿真计算，并且与其他一些方法的优化结果进行比较，结果表明该算法具有收敛速度快、计算精度高的突出优点。

刘佳等[4]提出了一种自适应 PSO 算法。该算法利用种群多样性信息对惯性权重进行非线性的调整，并在算法的后期引入速度变异算子和位置交叉算子，使算法摆脱后期易于陷入局部最优点的束缚。对基于向量评价的 PSO 算法进行了扩展，提出了基于向量评价的自适应 PSO 算法（Vector Evaluated Adaptive Particle Swarm Optimization，VEAPSO）来解决多目标无功优化问题，求解出问题的 Pareto 最优解集。为帮助决策者从 Pareto 最优解集中选取合适的最优解，刘佳等提出一种基于决策者偏好及投影寻踪模型的多属性决策法，使决策结果更加真实可靠。IEEE 30 和 IEEE 118 节点系统算例仿真表明该方法用于解决多目标无功优化问题是有效可行的。

刘自发等[5]提出了一种自适应小生境 PSO 算法，以粒子的位置状况及其 2 个向量点积的符号动态生成小生境半径，根据各粒子之间的距离组成小生境种群，在粒子群体优化过程中采用小生境技术，有效提高了算法的全局寻优能力，通过几个 IEEE 典型测试系统的计算及结果分析表明，该算法在解决电力系统无功优化问题时具有快速、高效、稳定的优点，为解决此问题提供了一种新的思路。

近年来，随着国民经济的快速发展，能源消耗日益增长，以集中式单一供电方式为主要特征的电力系统所引起的环境问题、能源问题、稳定性问题，以及经济性等问题[1]越来越引起人们的重视，进一步促使诸如微型燃气轮机、太阳能发电、风力发电、燃料电池等各种形式的分布式电源的发展和应用。由于分布式电源具有诸如能够安装在负荷中心、低投资成本、能够及时跟踪负荷变化，以及提高了系统的暂态稳定和电压稳定等优点[2~5]而具有很好的应用前景。当分布式电源（Distribution Generation，DG）接入配电网后，各支路的电流方向将发生改变，随之将引起配电网网络损耗的变化，使得网络损耗不仅与负荷大小有关，同时还与 DG 的安装位置以及分配到各节点的功率值有关，网络损耗随着分布式电源容量大小及位置的不同而不同。求解分布式电源选址和定容的问题属于求解多变量可行解问题，虽然也可以通过列举各种可行的方案来解决，但相当困难。近年来，国内外众多学者在这方面做了较多的研究工作。刘波等[6]在分析了分布式电源接入配电网前后网络损耗的变化情况的基础上，提出采用混合模拟退火（Simulated Annealing，SA）的改进 PSO 算法，对分布式电源选址和定容问题进行优化求解，并将计算结果与采用遗传算法、模拟退火算法的计算结果进行比较，验证了所提算法具有良好的收敛性和适应性。

配电网中补偿电容器的优化配置问题涉及如何最优地确定电容器的安装位置、容量及类型等方面。合理的电容器配置，能够有效改善电网的电压水平，降低系统的有功网损，并同时避免谐波谐振或电流放大。一直以来，电容器的优化配置是工程界和学术界所共同关注的研究课题。随着电力电子装置的应用日益广泛，谐波污染日趋严重，不合理的电容器配置可能导致某次或某几次频率下的谐波谐振或电流放大，从而危及电容器本身以及其他电气设备的安全运行。为了避免发生谐波谐振或电流放大，优化计算时有必要考虑谐波对电容器配置方案的影响。判断电网有无发生谐波放大和衡量电网畸变严重程度的主要指标是电压总谐波畸变率，各个国家对不同电压等级下的谐波畸变率都作了明确的限值规定。余欣梅等[7]建立了考虑谐波影响的电容器优化配置问题的非线性整数规划模型，并应用 PSO 算法进行求解，针对问题中的离散控制量对算法作了离散化处理，解决了 PSO 算法用于整数优化或混合整数优化的问题。对 IEEE69 节点系统进行了计算，结果表明算法是正确和有效的，可以获得电容器优化配置问题的全局最优解。

电力变压器是电力系统中的重要设备，其内部潜伏性故障的诊断对于确保整个电力系统安全可靠运行有着重要的意义。而油中溶解气体分析技术是目前对油浸变压器进行故障诊断最方便、有效的手段之一，能较准确、可靠地发现逐步发展的潜伏性故障,防止由此引起重大事故。王晓霞等[8]提出了一种改进的 PSO 算法，用来优化 BP 神经网络参数，并将其应用于电力变压器的故障诊断，该算法结合了 PSO 算法和 BP 算法的优点，具有训练快速和全局收敛的优点。贾嵘等[9]提出了一种邻域 PSO 算法优化基于 BP 神经网络的电力变压器油中气体分析（DGA）方法，即通过相关统计分析和数据的预处理，选择变压器油中典型气体作为神经网络的输入，然后利用训练好的邻域 PSO 算法优化后的神经网络进行变压器故障类型诊断。试验结果表明，该类方法具有很好的分类效果，较好地解决了变压器放电和过热共存时故障的难分辨问题，对故障类型的正判率较高。费胜巍等[10]提出了粒子群优化的支持向量机（PSO-SVM）变压器故障诊断方法，其中应用 PSO 算法进行 SVM 参数优化。采用 156 例变压器历史状态数据比较人工神经网络、SVM、PSO-SVM 的训练和诊断结果，其中 74 例用于模型的训练，82 例用于测试。试验结果表明，PSO-SVM 的诊断精度高于其他的诊断精度。

5.3 化工过程控制

化工领域的过程设计、生产控制、配方和计划等方面，在考虑产品性能、单位成本、环境影响等多因素下，通过数学建模最终成为多目标优化问题。多目标优化问题的求解是化工优化的重要组成部分。

（1）发酵过程是个间歇过程，其机理复杂，具有高度非线性、时变性。由于缺乏对发酵过程生物参数的有效检测技术，使得对其的优化操作、控制成为难点。熊伟丽等[11]采用 PSO 算法实现对参数的同时寻优，并以β-甘露聚糖酶发酵过程产物浓度为对象，建立基于 PSO-SVR 的状态预估模型。通过实际发酵应用表明：该模型具有很好的学习精度和泛化能力，可实现对β-甘露聚糖酶产物浓度的实时在线预估。

（2）随着化学工业的发展，以间歇、半间歇过程作主要生产方式的精细化工在国民经济中的地位越来越重要。所谓精细化工（Fine Chemistry）一般指产品具有批量小、纯度或质量要求高、附加值高等特点的化学工业，如药品、高分子、生物工程、涂料、香料、试剂、化妆品等。由于精细化工产品品种多、市场变化快，因此适用于间歇（及半间歇）生产。间歇生产的主要特点为有限生产和无现实稳态性，所谓有限生产指间歇生产的每个批次持续的时间是有限的，如链霉素发酵的时间一般为 200h 左右。而无现实稳态性指间歇生产过程不同于连续生产过程，属于非稳态

生产操作，反应器内物料的浓度及反应速率等均随时间变化。间歇生产过程一般都有特定的生产方案（Recipe）作为指导，过程的重要变量如温度、压力、原料浓度等均有工艺曲线作为参考，生产方案的目的在于提高间歇生产过程的可重复性，保证批次间产品质量的一致性。但是由于过程本身的复杂性以及干扰的影响，实际操作条件往往会和生产方案有所区别，并最终导致产品质量的下降。实施间歇过程监控的目的在于及时发现生产过程的非正常情况、提高生产安全性、降低生产成本、及时发现并排除生产故障，提高产品质量的一致性。统计过程监控基于历史正常生产数据，使用数理统计的理论建立正常工况下操作条件的统计监控模型，并用于实际生产的在线监控。作为数据驱动的监控方法，统计过程监控不需要任何过程的机理模型，因此用途十分广泛。谢磊等[12]结合鲁棒估计理论的数据处理优势，提出一种新的基于 PSO 算法的鲁棒 MPCA 算法，给出了鲁棒统计过程监控统计量的计算方法，用于处理建模数据中存在离群点时的间歇生产过程监控问题。为比较传统 MPCA 算法和鲁棒 MPCA 算法监控的效果，还对某制药厂的 25 批产品效价在 25000~30000 之间的、具有正常变化趋势以及两批具有异常初始 pH 值（作为离群点）的链霉素发酵的批报数据分别建立了传统 MPCA 算法和鲁棒 MPCA 算法模型，实际链霉素生产数据的应用表明，鲁棒 MPCA 算法能够有效克服离群点的影响，保证统计模型不过分依赖建模数据，有效减少了模型对数据的要求。

（3）物料平衡是氧化铝生产的核心，通过物料平衡计算可以选出最佳的生产方法和工艺流程，从而达到投资最省及生产成本最低的目的。目前全世界氧化铝和氢氧化铝生产 90 %以上是用拜耳法，但拜耳法生产工艺非常复杂，因而导致了其物料平衡计算的复杂。通过对其工艺分析、建模，发现可以将其归结于非线性多目标约束优化问题的求解，但计算量大，传统的计算方法与计算工具不仅需耗费较长的时间，也容易陷入局部最优解。孔力等[13]给出了一种具有随机变异特性的 PSO 算法，并应用于拜耳法物料平衡计算。计算结果表明：新算法不仅在稳定性和收敛性上优于基本 PSO 算法，且具有较高的全局收敛能力，是进行拜耳法物料平衡计算的一种行之有效的方法。

（4）液液平衡和汽液平衡计算十分重要，它是萃取、萃取精馏和非均相共沸精馏等分离过程设计和优化的基础。Henley 和 Rosen 最先提出三相等温闪蒸的计算方法，发现体系中存在两个部分互溶液相时，液相的非理想性将使 K 值对液相组成相当敏感，令算法难以收敛。随着优化方法的进步，通过体系 Gibbs 自由能最小化，含两液相的相平衡计算的算法也在不断发展。基于牛顿算法的优化算法只能实现局部优化，相平衡则需实施全局优化，才能求得体系的真解。成飙等[14]对复杂相平衡问题进行了分析，将原命题转换为受立方型可行域约束的最优化问题，并采用改进

的 PSO 算法，全局最小化 Gibbs 自由能，从而实现复杂相平衡求解。

（5）丙烯腈是一种重要的有机化工原料，在合成纤维、合成脂、合成橡胶等高分子材料领域有广泛的应用，丙烯腈聚物与丙烯腈衍生物也有重要的用途。在丙烯腈装置的生产过程中，丙烯腈收率是一个关键指标，及时、准确地测量产品中丙烯腈收率是进行丙烯腈装置进程控制的关键。在实际生产中，由于缺乏在线分析仪表，以常用人工采样分析的方法得到丙烯腈收率，通常情况下采样到分析再到最终结果要经过几个小时的时间，即使不计较价格的昂贵与维护保养的复杂性而采用在线分析仪，其测量的滞后也难满足生产要求。陈国初等[15]对 PSO 算法与模糊神经网络的结合进行研究，提出粒子群模糊神经网络，并将其应用于丙烯腈收率测量建模。该方法采用模糊神经网络来构建丙烯腈收率软测量模型，用 PSO 算法优化模糊神经网络的参数，并结合实际工艺，对所建测量模型进行仿真研究。实验结果表明，该模型的性能优于其他模型，能够准确预测丙烯腈收率，具有较高的精度和良好的应用前景。

（6）换热器是进行热量交换的通用工艺设备，广泛应用于化工、炼油、动力、冶金、原子能等工业部门，是生产与生活中不可缺少的重要热能利用设备，其性能的优劣对能源的有效利用会产生重大的影响。因此在实际生产中针对换热器的某一个或者几个性能进行优化设计具有重要意义。换热器的优化设计，就是要求所设计的换热器在满足一定要求的条件下，使其一个或数个性能指标达到最好。换热器的优化方法有很多种，究竟选用什么优化方法，常常靠经验、靠摸索。韩武涛等[16]以内外翅片管换热器为例，利用 PSO 算法来进行换热器的优化设计。结果表明，内外翅片管换热器以体积为优化目标时，经过 PSO 算法优化后，虽然总管数比用遗传算法优化时要多，但管子横向间距和管外翅片间距都缩小，在这种情况下阻力也减小，其原因在于此时气体流动方向的长度缩短了，换热器更加紧凑，换热面积也大幅减少，重量也减轻了。

5.4 机器人领域的应用

（1）路径规划是移动机器人导航的最基本环节一，是按照某一性能指标搜索一条从起始状态到目标状态的最优或近似的无碰撞路径。根据机器人对环境信息掌握的程度，可以分为两种类型：环境信息完全已知的全局路径规划和环境信息完全未知或部分未知的局部路径规划。对于环境信息完全已知的情况，到目前已经有许多解决方法，例如势场法、可视图法等。势场法结构简单，易于实现，得到了广泛的应用，但也有较大缺陷：

① 陷阱区域；
② 相近障碍物之间难发现路径；
③ 在障碍物前易震荡；
④ 在狭窄通道中摆动。

秦元庆等[17]提出一种路径规划的新方法，用自由空间法建立规划环境模型，将规划分为两个层次：用图论方法寻求一条无碰撞次优路径；用 PSO 算法优化次优路径，得到全局最优路径，仿真结果取得很好效果。成伟明等[18]针对传统神经网络方法进行路径规划时对每个障碍均设计一些特定的隐节点，当障碍较多且环境为动态时，存在网络结构庞大且神经元的阈值随时间的变化而需要不断改变的缺点，在对已有方法研究的基础上，结合小波网络和径向基函数网络的优点，采用一种新的四层的网络进行路径规划。以机器人当前位置作为网络的输入，根据传感器获得的环境障碍信息，利用 PSO 算法无交叉、快速、易操作的优点训练网络权值，网络的输出是下一时刻的机器人位置。提出了一种新的能量函数作为粒子群的适应度函数，利用函数各部分之间相互制约的作用，引导机器人朝目标方向运动，避开障碍的同时，使路径尽量较优。最后进行了仿真实验，实验结果验证了方法的有效性。薛英花等[19]针对常规路径规划单纯追求路径最短、路径规划不灵活和实现复杂的缺点，提出了一种改进的移动机器人全局路径规划方法。该方法综合人工势场（APF）与 PSO 算法的优点，利用障碍物的排斥力生成路径的危险度地图（DDM），将路径长度与危险度的加权和作为 PSO 算法的适应度函数，获得了一条全局最优路径。该方法具有 3 个优点：粒子初始化及更新过程中会自动避开有障碍物的危险区域，规划出一条既安全相对长度又较短的路径；通过调整加权因子平衡长度与危险度在适应度函数中的比重，路径规划灵活；算法实现简单，收敛速度快，能满足移动机器人实时路径规划的要求。仿真结果证明了该算法的可行性和有效性。邓高峰等[20]针对机器人在障碍环境下寻找最优路径问题，提出了一种障碍环境下机器人路径规划的蚁群 PSO 算法。该方法有效地结合了 PSO 算法和蚁群算法的优点，采用栅格法进行环境建模，利用 PSO 算法的快速简洁等特点得到蚁群算法初始信息素分布，以减少迭代次数，加快算法的收敛速度；同时利用蚁群算法之间的可并行性，采用分布式技术实现蚂蚁之间的并行搜索，提高求解精度高。仿真实验结果证明了该方法的有效性，是机器人路径规划的一种较好的方法。

（2）张皓等[21]针对 PSO 算法对约束条件的优化处理问题，提出一种具有自适应度双群体 PSO 优化算法，该算法将目标函数与约束条件分别考虑，形成 2 种群体以不同目标为前提同时向最优解进化；并分别对 2 种群体的适应度引入自适应权重系数与相应调整策略，基于并非所有非可行个体均劣于可行个体的概念，动态地调整

其适应度以保证部分非可行个体向可行域进化。将其应用于组群机器人队形控制中，链型结构(纵队)队形仿真结果表明了该算法的有效性。该算法为实际应用中约束优化问题的求解提供了新的途径。

（3）水下机器人是一个多变量、强耦合、高度非线性的动力学系统。由于其本身独有的动力学特性、水下环境的不确定性和任务的复杂性，给控制该对象带来很多挑战。唐旭东等[22]提出一种带可变学习因子和惩戒因子的改进的 PSO 算法，并将该算法应用于水下机器人 S 面控制器参数的整定优化中，进行仿真试验研究，结果验证了该方法的有效性。徐贺等[23]针对一种可重构轮式移动机器人扩展构型中重构变量间的非线性关系，采用解析结合投影几何的方法建立可重构的移动机器人的稳定性指标模型。运用 PSO 算法获得了在等效稳定裕量最大前提下的移动机器人重构模型，其结果同移动机器人重构实验的结果吻合很好。

（4）目标的识别与精确定位是工业机器人完成各项操作任务的关键技术之一。在复杂工业环境下，机器人感知检测系统需同时使用 CCD 摄像机与激光扫描仪，这两种传感器数据具有很好的互补性，对二者测量数据进行融合将获得更为精确的目标 3D 信息。摄像机与激光扫描仪的外参数标定，即估计两者位姿关系，是摄像机与激光扫描仪信息融合的基础。汪霖等[24]提出一种新的摄像机与激光扫描仪外参数自标定方法，该方法无需特定的标定模板，通过云台俯仰运动，调整激光扫描仪的扫描角度，改变机器人工作台面内激光扫描线位置，可获得不同位置的摄像机和激光扫描仪测量数据；利用激光扫描点在图像上的投影线约束条件，建立外参数的非线性优化函数。采用 PSO 算法搜索非线性优化函数的全局极小点，完成外参数优化。PSO 算法优化过程中，首先建立外参数和粒子群之间的对应关系，再根据粒子的适应度函数取值不断更新其速度和位置，直到获得最优解。PSO 算法避免了初值选取问题，只需确定外参数的粗略范围。实验结果表明这种算法具有较强的寻优能力，标定结果精度高。

5.5 计算机工程领域应用

（1）由于 Web 上积累了大量的 XML 数据，形成了许多“脏数据”，阻碍了商业应用，因此需要对它们进行挖掘、清洗，以保证和提高数据质量。国外信息化程度较高，对数据清洗的研究较多；国内的数据清洗研究集中在：

① 特殊域清洗，主要解决某类特定应用域的数据清洗，用一个大的预定义规则库处理清洗过程中发现的问题，是目前研究较多的领域；

② 与特定应用领域无关的数据清洗，主要集中在清洗重复的记录。

这些研究或多或少存在不完备的地方，主要表现在：

① 研究主要集中在字符型数据上；

② 检测重复记录遭遇海量数据时，耗时太多，检测效率与精度较差；

③ 大多数数据清洗工具都是针对特定领域的，其应用受到一定的限制；

④ 研究主要集中在结构化数据上。

针对以上问题，翟学敏等[25]以 XML 键为基础讨论构建 XML 数据清洗的过程：首先量化 XML 文档，利用多模板隐马尔可夫算法（Hidden Markov Model，HMM）提取 XML 键值信息，利用波函数方法对 PSO 算法进行优化，对相似 XML 数据查找定位，然后完成合并、剔除、转换、补充等操作。研究了半结构化 XML 键的选取、知识库的抽取、量化计算方法与 PSO 算法的融合等问题。研究结果表明，随着记录规模的扩大，提出的算法在时间性能上具有明显的优势。

（2）测试数据自动生成是软件测试过程中一个关键的问题。现有的结构测试数据自动生成，多采用基于遗传算法的方法。这些方法存在算法复杂、参数不易设置等问题。李爱国等[26]提出一种基于 PSO 算法的软件结构测试数据自动生成方法，以分支函数叠加法作为适应度值函数。实验结果表明，与基于遗传算法的方法相比，可以更高效地生成测试数据，在粒子数目与种群个数相同的情况下生成所需测试数据的迭代次数仅是遗传算法的 1/16 左右。

（3）针对现有服务组合中 QoS 优化的不足，刘莉平等[27]提出一种基于 PSO 算法的解决 QoS 动态服务组合的算法。通过对服务组合的业务逻辑与服务实例进行合理编码，重新定义粒子的位置、速度与“加”运算，利用 PSO 算法的智能优化原理以及局部与全局优化信息加快粒子群的搜索速度，使其能够快速地得到一组满足约束条件的 Pareto 优化的服务组合。实验结果证明了算法的可行性和有效性。夏虹等[28]针对现有 Web 服务组合中服务选择技术的不足，提出了一种基于 PSO 算法的多目标优化策略，用于解决 Web 服务组合中基于服务质量的服务选择全局最优化问题。将 Web 服务选择全局最优化问题转化为一个带 QoS 约束的多目标服务组合优化问题，利用多目标 PSO 算法的智能优化原理，通过同时优化多个 QoS 参数，最终产生一组满足约束条件的 Pareto 最优解。实验结果证明了该算法的可行性和有效性。

（4）入侵检测系统是一种主动的安全防护系统，通过对网络流量或主机运行状态的检测来发现对系统资源的非授权访问与破坏行为，并对各种恶意攻击作出响应。入侵检测中的异常检测模型假定所有的入侵活动都是异常活动，由于在合理的尺度条件下属于同样的类别（攻击类型或正常类型）的数据实例在特征空间中互相接近，而不同类别的数据实例则相互远离，因此可以将聚类算法应用于入侵检测系统。目

前普遍采用的模糊均值聚类的主要缺点是对初始值过于敏感，算法易收敛到局部最优解。唐贤伦等[29]利用 PSO 算法全局寻优、快速收敛的优点结合 FCM 算法提出基于 PSO 算法和模糊均值聚类的入侵检测方法。

（5）陆锦军等[30]为了改善网络拥塞控制系统的性能，基于流体理论的网络简化模型，将量子空间中的 PSO 算法应用于 PID 控制器参数优化，定义了一个综合调节时间、上升时间、超调量、系统静态误差、正弦跟踪误差等动静态性能指标函数，在给定的参数空间进行组合优化搜索，迅速求得获取使性能指标优化函数极小化的一组 PID 控制器参数，将 PID 控制器应用于网络主动队列管理系统中。仿真结果表明，在大时滞和突发业务流冲击的两种情况下，该方法设计的控制器的动静态性能优于 REDPI 算法，也优于遗传算法的优化结果，超调量均小于 4%，调节时间均小于 4s，稳态误差均小于 2 个数据包。

（6）随着 Web 技术及其应用的快速发展，XML 文档已经成为互联网上信息表示和数据交换的一个重要标准，其作用已深入到网络社区的每个角落。刘波等[31]针对 XML 文档进行群体搜索的特点与不足，提出利用群智能算法的概率变换规则对其进行改进，首先采用路径离散化规则，结合 XML 半结构化的特点及概率知识，再融合 PSO 算法与蚁群算法进行动态群体搜索，而群体自适应杂交、多次编码、迭代选择等不仅可以提高数据搜索的范围、精度和收敛的效率，而且可以避免早熟，降低算法的复杂度。仿真实验表明这种融合方法具有更好的查询效果。

5.6 通信工程领域应用

（1）董伟等[33]针对多输入多输出系统的联合频偏信道估计问题，考虑了更一般的模型（每一根发送天线到每一根接收天线之间的频偏是不同的），研究了频偏和信道的最大似然估计。分析表明，该估计问题包含一个多维搜索过程。为了解决上述复杂的估计问题，提出了一种联合频偏信道估计新方法，首先根据粒子群优化理论估计出多个发射天线到某一接收天线的频偏，然后再利用最大似然估计器对信道增益进行估计。仿真结果表明，与基于相关的估计算法相比，所提出的算法有更大的频偏估计范围，且估计值的均方误差渐近达到 Cramer-Rao 下界。

（2）当传输速率大于 10Gb/s 时，PMD（偏振模色散）对系统的损害就明显地表现出来，成为高速率光纤通信系统发展的主要障碍。以往人们对单信道 PMD 补偿进行了大量的研究，而 WDM（密集波分复用）系统多信道的 PMD 补偿仍然是一个难题。目前关于 WDM 系统 PMD 补偿方案主要有两种：一是将所有的信道解复用后单独进行 PMD 补偿，由于信道太多，补偿系统过于庞大，显然是不太经济的；另一

种是用一个补偿器补偿所有的信道，但这种补偿方案可能将最坏信道补好的同时恶化其他信道，因而控制算法的研究成为该方案的研究重点。张倩等[34]根据第二种多信道 PMD 补偿方案，将 PSO 算法用于多信道的 PMD 补偿，在补偿最坏信道的同时，使其他信道的性能也得到保障，并以两信道为例进行了数值模拟，取得了很好的补偿效果。

（3）PSO 算法广泛用于解决通信系统的多用户检测问题。多载波码分多址（MC-CDMA）系统将多载波传输技术与 CDMA 技术相结合，具有系统容量大，频谱利用率高，抗符号间干扰（ISI）等优点，成为下一代移动通信核心技术之一。在 MC-CDMA 系统中，用户符号经扩频后的码片调制到不同的子载波，不同用户符号在不同子载波上经历了相互独立的衰落，破坏了扩频码间的正交性，故在信号接收端仍需采用多用户检测技术获得不同用户的信号。由于 MC-CDMA 系统中接收端经 DFT 解调输出的信号与 CDMA 系统接收的信号有相似的数学模型，所以 MC-CDMA 系统的多用户检测可引用 CDMA 多用户检测的方法。在研究 CDMA 多用户检测问题时已知，最佳多用户检测是一个全局寻优的过程，其算法的复杂性和用户个数成指数关系，计算量大，在实际应用中难以实时实现。典型的次最佳多用户检测，如解相关多用户检测和 MMSE 多用户检测等，在一定程度上消除了多址干扰，但都存在着一些不足。鉴于离散 PSO 算法在解决 CDMA 多用户检测问题时具有较快的收敛速度、较好的误码性能、较低的计算复杂度和较少的参数设置，龙银芳等[35]将离散 PSO 算法用于解决 MC-CDMA 多用户检测问题。通过仿真验证了将其应用于 MC-CDMA 系统的多用户检测的可行性及其良好的性能，并将该法的性能与传统的最小均方误差（MMSE）多用户检测及最佳多用户检测（OMD）的性能进行了比较。

（4）空时分组码-多载波-码分多址（STBC-MC-CDMA）系统将 STBC 与 MC-CDMA 相结合，使宽带系统获得了空间分集增益。在 STBC-MC-CDMA 系统上行链路中，由于多个用户的信号在不同子载波上经历了相互独立的衰落，从而破坏了不同用户特征序列的正交性，造成了比较严重的多址干扰（MAI）。因此，基站接收端需要采用先进的信号处理技术来获得不同用户的信号。由于传统的最优化多用户检测(MUD)的复杂度随用户数目的增加呈指数增长，难于付诸实时应用，因此，研究各种具有良好性能和较低复杂度的次优 MUD 方案具有重要意义。刘洪武等[36]研究了使用多天线分集接收的 STBC-MC-CDMA 系统上行链路中基于 PSO 算法的 MUD。算法对不同天线分支的匹配度函数按 Pareto 准则进行优化并更新粒子速度和位置，独立利用了不同天线分支信号携带的有用信息。仿真结果表明，所提方案独立利用了不同天线分支的信道信息，获得了增强的开发和探索能力，其性能优于常规 PSO 算法和多目标遗传算法。

（5）光纤通信传输信息容量主要受到损耗与色散两个因素的限制，单模光纤中的色散主要包括色度色散（CD）和 PMD。随着各种色度色散补偿技术不断完善，在传输速率大于 10Gbps 的光纤通信系统中，PMD 补偿问题越来越受到人们的重视。国外已有多种 PMD 自适应补偿实验系统研制成功。由于实验条件限制，国内目前对 PMD 补偿的理论分析工作较多。张晓光等[37]研究成功了 PMD 自适应补偿实验系统，实验中采用特定频率分量功率取样，并采用了 PSO 算法作为反馈控制算法，实现了跟踪时间为 1~2 s、PMD 补偿量达 30 ps 的自动跟踪补偿。

（6）近年来，无线传感器网络由于在目标探测和跟踪、环境监测、工业过程监控以及军事应用上具有广泛的应用前景，备受研究者瞩目。节点定位是无线传感器网络(Wireless Sensor Networks，WSN)的关键应用支撑技术，对于大多应用场合来说，带有位置信息的传感数据才具有实际意义。然而 WSN 节点的计算和通信能力都十分有限，而且大量节点的部署往往无法人为控制，因此设计高效的节点定位算法就显得十分重要。王驭风等[38]提出一种集合了 Range-free 算法、测距技术和 PSO 算法的节点定位综合算法，并将其应用于移动节点定位，通过仿真验证了该算法具有优良的定位性能，使 range-free 算法的定位误差大幅下降，节点的通信和计算损耗并没有急剧增加而在可接受的范围内，同时在稀疏网络中也具有良好的校正效果。

（7）节能设计是无线传感器网络设计的关键，然而，无线通信模块空闲状态和接收状态能耗接近，睡眠状态能耗最少。因此，无线传感器网络必须利用节点工作状态的转换，使节点在网络正常运作下尽快进入睡眠状态，关闭通信模块，高效使用能量，延长网络生命周期。从协议设计角度看，高效的节点 MAC 层休眠/唤醒调度机制对无线传感器网络非常重要。众所周知，TDMA 较基于竞争的 MAC 协议具有限制传输时延和避免冲突的优点。在高数据传输速率冲突易发的网络环境下，发生碰撞后的数据重发容易产生时延和额外的能量消耗。因此，在如节点剩余能量监测、多目标定位或区域环境监测等需要周期性全局快速数据收集的应用中，使用 TDMA 调度算法能获得更好的运行效果。丁英强等[39]针对无线传感器网络节点能量有限的特点，提出了一种应用于簇结构的 MAC 协议（FT-MAC）。该协议在簇内采用分时通信，并通过 PSO 算法优化簇内时隙分配，减少节点状态转换的能量消耗。为了消除簇内通信产生的簇间干扰，FT-MAC 向网络中各簇分配了不同的频率。与其他分时协议相比，FT-MAC 利用各簇网关节点的特殊工作模式解决了簇间通信问题，使网络只需保持簇内时间同步，同时避免了全网时间同步的大量能耗。仿真结果表明，FT-MAC 协议可以有效减少传感器网络的能耗，并且具有较小的数据包延迟时间，在具有簇状结构的大规模无线传感器网络中具有广阔的应用前景。

参 考 文 献

[1] 刘连志，顾雪平，刘艳．不同黑启动方案下电网重构效率的评估．电力系统及自动化，2009，33(5)：24～28．

[2] 韩恺，赵均，钱积新．电力系统机组组合问题的闭环粒子群算法．电力系统及自动化，2009，33(1)：36～69．

[3] 赵波，曹一家．电力系统无功优化的多智能体粒子群优化算法．中国电机工程学报, 第 25 卷第 5 期：1～7.

[4] 刘佳，李丹，高立群，宋立新．多目标无功优化的向量评价自适应粒子群算法．中国电机工程学报，第 28 卷第 31 期：22～28.

[5] 刘自发，张建华．基于自适应小生境粒子群优化算法的电力系统无功优化．电力自动化设备，第 29 卷第 11 期，2009（11）： 27～30.

[6] 刘波，张焰，杨娜．改进的粒子群优化算法在分布式电源选址和定容中的应用．电工技术学报，第 23 卷第 2 期，2008（2）：103～108.

[7] 余欣梅，李妍，熊信良，吴耀武．基于 PSO 考虑谐波影响的补偿电容器优化配置．中国电机工程学报，第 23 卷第 2 期，2003（2）：26～120.

[8] 王晓霞，王涛．基于粒子群优化神经网络的变压器故障诊断．高电压技术，第 34 卷第 11 期，2008（11）：2362～2367.

[9] 贾嵘，徐其惠，李辉，刘伟．基于邻域粒子群优化神经网络的变压器故障诊断．高压电器，第 44 卷第 1 期，2008（2）：8～19.

[10] 费胜巍，苗玉彬，刘成良，张晓斌．基于粒子群优化支持向量机的变压器故障诊断．高电压技术，第 35 卷第 3 期, 2009（3）：509～513.

[11] 熊伟丽，徐保国．基于 PSO-SVR 的发酵过程状态预估模型．南京理工大学学报（自然科学版），32 卷第 4 期，2008（8）：517～521.

[12] 谢磊，王树青，张建明．基于 PSO 的间歇生产鲁棒统计过程监控．化工学报，第 56 卷第 3 期，2005（3）：492～498.

[13] 孔力，程晶晶，宋胜利，苏日建．基于改进粒子群优化技术的拜耳法物料平衡计算．华中科技大学学报（自然科学版），第 36 卷第 1 期，2008（1）：95～98.

[14] 成飙，陈德钊．基于混合粒子群算法的复杂相平衡计算方法．高校化学工程学报，第 22 卷第 2 期，2008（4）：320～323.

[15] 陈国初，徐余法，俞金寿．基于粒子群模糊神经网络的丙烯腈收率软测量建模．系统仿真学报，第 19 卷第 23 期，2007（12）：5370～5372.

[16] 韩武涛，谢公南，曾敏，王秋旺．基于粒子群算法的内外翅片管换热器优化．高校化学工程学报，第 22 卷第 5 期，2008（10）：744～749．
[17] 秦元庆，孙德宝，李宁，马强．基于粒子群算法的移动机器人路径规划．机器人，第 26 卷第 3 期，2004（5）：222～225．
[18] 成伟明，唐振民，赵春霞，陈得宝．基于神经网络和 PSO 的机器人路径规划研究．系统仿真学报， 第 20 卷第 3 期，2008（2）：608～611．
[19] 薛英花，田国会，李国栋．利用改进的粒子群算法的机器人全局路径规划．华中科技大学学报（自然科学版），第 36 卷增刊，2008（10）：167～170．
[20] 邓高峰，张雪萍，刘彦萍．一种障碍环境下机器人路径规划的蚁群粒子群．控制理论与应用，第 26 卷第 8 期，2009（8）：879～883．
[21] 张皓，陈雪波，马德楠．具有自适应度双群体 PSO 的组群机器人队形控制．清华大学学报（自然科学版）2008 年第 48 卷第 52 期：1751～1755．
[22] 唐旭东，庞永杰，万磊．改进 PSO 算法在水下机器人 S 面运动控制参数整定中的应用．应用基础与工程科学学报，第 17 卷 1 期，2009（2）：153～160．
[23] 徐贺，封海波，栾钰琨，彭高亮，高晓智，武永见．基于粒子群算法和投影解析法的机器人构型优化．哈尔滨工程大学学报，第 30 卷第 8 期，2009（8）：918～923．
[24] 汪霖，曹建福，韩崇昭．基于粒子群优化的机器人多传感器自标定方法．机器人，第 31 卷第 5 期，200（9）：391～396．
[25] 翟学敏，刘渊，刘波，毕蓉蓉．改进的 XML 智能数据清洗策略．计算机工程，第 35 卷第 4 期，2009（2）：66～71．
[26] 李爱国，张艳丽．基于 PSO 的软件结构测试数据自动生成方法．计算机工程，第 34 卷第 6 期，2008（3）：93～97．
[27] 刘莉平，陈志刚，刘爱心．基于粒子群算法的 Web 服务组合研究．计算机工程，第 34 卷第 5 期，2008（3）：104～112．
[28] 夏虹，李增智．粒子群算法求解 Web 服务组合中基于 QoS 的服务选择．北京邮电大学学报，第 32 卷第 4 期，2009（8）：63～67．
[29] 唐贤伦，庄陵，李银国，曹长修．基于粒子群优化和模糊 c 均值聚类的入侵检测．计算机工程，第 34 卷第 4 期．2008（2）：13～15．
[30] 陆锦军，王执铨．基于量子粒子群优化的网络拥塞控制新策略．东南大学学报（自然科学版），第 38 卷增刊(Ⅱ)，2008（11）：101～106．
[31] 刘波，杨路明，邓云龙．自适应的混沌粒子群算法优化 XML 文档聚类策略．系统仿真学报，第 21 卷第 3 期，2009（2）：716～720．

[32] 童亚拉．自适应动态演化粒子群算法在 Web 主题信息搜索中的应用．武汉大学学报（信息科学版），第 33 卷第 12 期，200（12）：1296～1299．
[33] 董伟，李建东，吕卓，贺鹏．MIMO 系统联合参数估计．西安电子科技大学学报（自然科学版），第 35 卷第 2 期，2008（4）：189～195．
[34] 张倩，段高燕，张晓光，蒋达娅．PSO 算法用于 WDM 系统多信道 PMD 补偿的研究．光子学报，第 37 卷第 9 期，2008（9）：1829～1832．
[35] 龙银芳，赵知劲，赵治栋．基于离散粒子群算法的 MC-CDMA 多用户检测．压电与声光，第 31 卷第 3 期，2009（6）：305～308．
[36] 刘洪武，冯全源．分集接收的 STBC-MC-CDMA 系统中基于 PSO 算法的多用户检测．电子与信息学报，第 31 卷第 1 期，2009（1）：45～48．
[37] 张晓光，于丽，郑远等．光纤通信系统中偏振模色散自适应补偿实验研究．光子学报，第 32 卷第 12 期，2003（12）：1474～1478．
[38] 王驭风，王岩．基于矢量的无线传感器网络节点定位综合算法．通信学报，第 29 卷第 11 期，2008（11）：227～236．
[39] 丁英强，孙雨耕，李婷雪．基于时分频分的无线传感器分簇网络 MAC 层协议．天津大学学报，第 41 卷第 8 期，2008（8）：904～910．

附录A　常用的基准测试函数

1．Sphere函数

该函数是单峰测试函数，主要用于测试算法的寻优精度，函数 3D 图形如图 A.1 所示，表达式为

$$f(X)=\sum_{i=1}^{n} x_i^2$$

搜索空间：$-100 \leqslant x_i \leqslant 100$。

全局最优：$f(x)=0$，$x_i=0$。

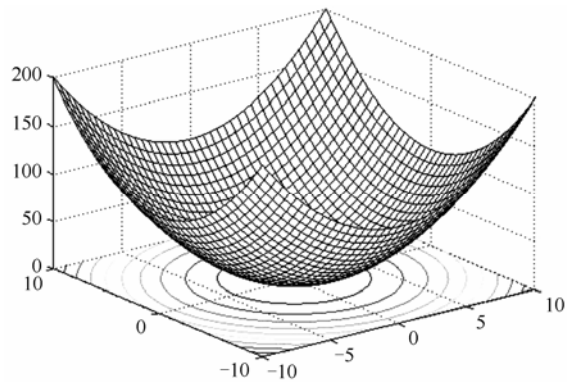

图 A.1　Sphere 函数的 3D 图形

2．Schwefel函数

该函数的全局最优具有欺骗性，常用来测试函数的收敛性能，函数 3D 图形如图 A.2 所示，表达式为

$$f(x)=\sum_{i=1}^{n}[-x_i \sin(\sqrt{|x_i|})]$$

搜索空间：$-500 \leqslant x_i \leqslant 500$。

全局最优：$f(x)=-418.9829$，$x_i=420.9687$。

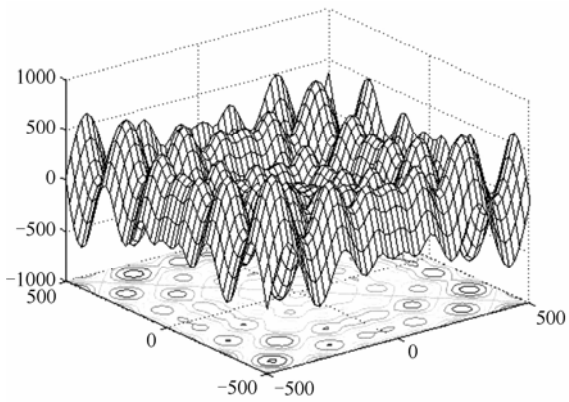

图 A.2　Schwefel's 函数的 3D 图形

3．Schwefel's Problem 1.2 函数

该函数是单峰、不可分离（Non-Separable）、可扩展函数，函数 3D 图形如图 A.3 所示，函数表达式为：

$$f(x)=\sum_{i=1}^{n}(\sum_{j=1}^{i}x_j)^2$$

搜索空间：$-100\leqslant x_i\leqslant 100$。

全局最优：$f(x)=0$，$x_i=0$。

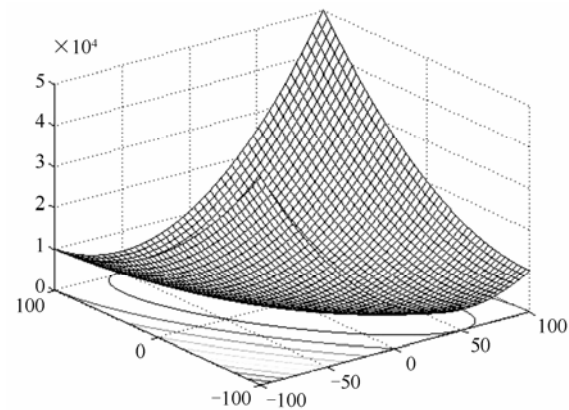

图 A.3　Schwefel's Fuction1.2 函数的 3D 图形

4．Schwefel’s Problem 2.22 函数

该函数是单峰测试函数，函数的 3D 图形如图 A.4 所示，函数表达式为

$$f(x)=\sum_{i=1}^{n}|x_i|+\prod_{i=1}^{n}|x_i|$$

搜索空间：$-10\leqslant x_i\leqslant 10$。

全局最优：$f(x)=0$，$x_i=0$。

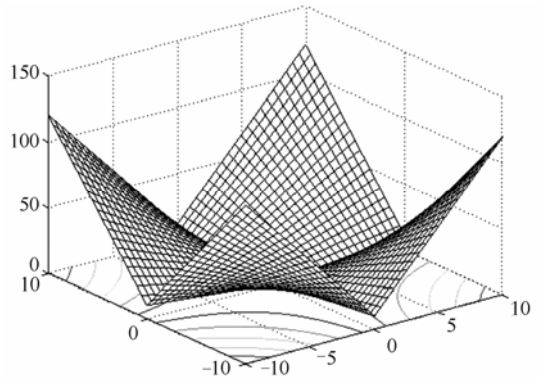

图 A.4　Schwefel’s Fuction2.22 函数的 3D 图形

5．Schwefel’s Problem 2.21 函数

该函数为单峰测试函数，函数的 3D 图形如图 A.5 所示，函数表达式为

$$f(x)=\max_i\{|x_i|,1\leqslant i\leqslant n\}$$

搜索空间：$-100\leqslant x_i\leqslant 100$。

全局最优：$f(x)=0$，$x_i=0$。

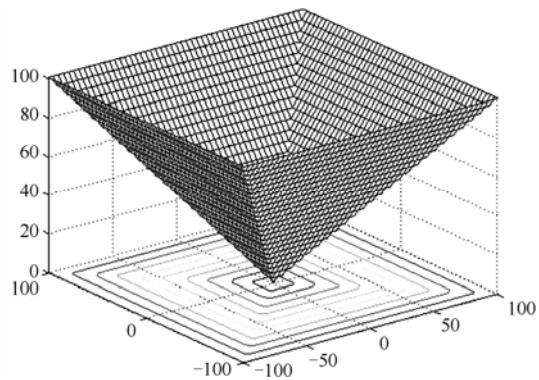

图 A.5　Schwefel’s Problem 2.21 函数的 3D 图形

6．Rosenbrock函数

该函数为多峰测试函数，在局部最优和全局最小之间有一个非常狭窄的波谷，常用来测试算法的执行性能，3D 图形如图 A.6 所示，表达式为

$$f(x)=\sum_{i=1}^{n-1}(100(x_i^2-x_{i+1})^2+(x_i-1)^2)$$

搜索空间：$-100\leqslant x_i\leqslant 100$。

全局最优：$f(x)=0$，$x_i=0$。

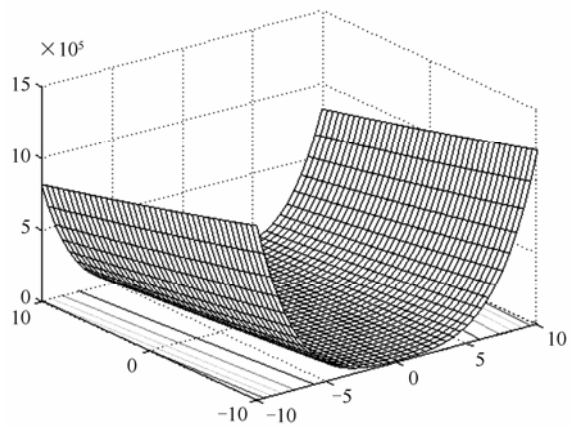

图 A.6　Rosenbrock 函数的 3D 图形

7．Ackley函数

该函数为多峰测试函数，函数的 3D 图形如图 A.7 所示，函数表达式为

$$f(x)=-20\mathrm{e}^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}}-\mathrm{e}^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)}+20+\mathrm{e}$$

搜索空间：$-50\leqslant x_i\leqslant 50$。

全局最优：$f(x)=0$，$x_i=0$。

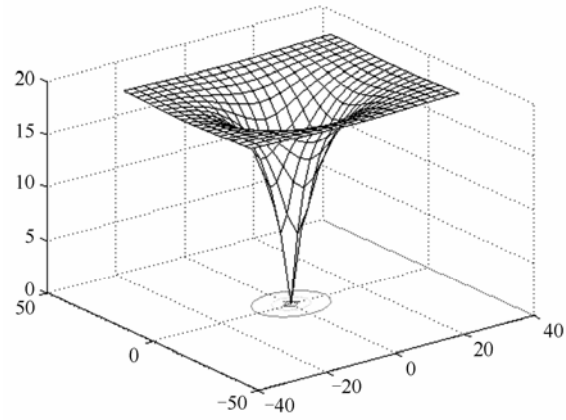

图 A.7 Ackley 函数的 3D 图形

8．Griewangk函数

和 Rastrigin 函数相似，多个极小值呈规律性分布，3D 图形如图 A.8 所示，局部 3D 图形如图 A.9 所示，函数表达式为

$$f(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2+\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$$

搜索空间：$-600\leqslant x_i\leqslant 600$。

全局最优：$f(x)=0$，$x_i=0$。

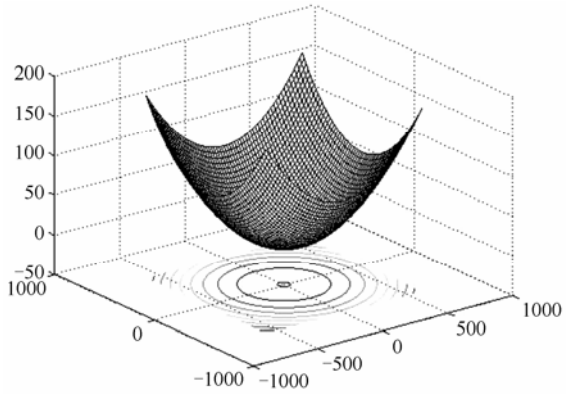

图 A.8 Griewangk 函数的 3D 图形

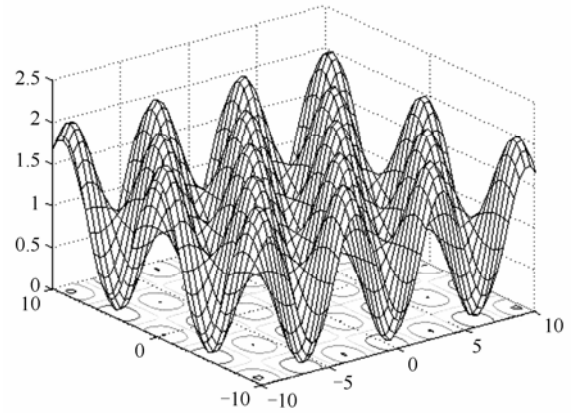

图 A.9　Griewangk 函数局部 3D 图形

9．Rastrigin函数

该函数为多峰测试函数，极小值分布呈现规律性，函数 3D 图形如图 A.10 所示，函数表达式为

$$f(x)=100n+\sum[x_i^2-10\cos(2\pi x_i)]$$

搜索空间：$-10\leqslant x_i\leqslant 10$。

全局最优：$f(x)=0$，$x_i=0$。

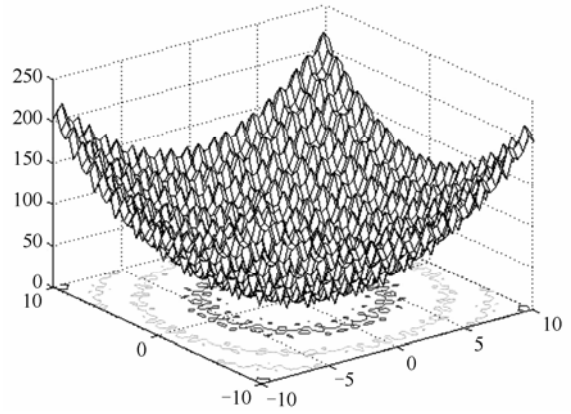

图 A.10　Rastrigin 函数 3D 图形

10．Weierstrass函数

该函数为多峰测试函数，函数的 3D 图形如图 A.11 所示，表达式为

$$f(x)=\sum_{i=1}^{n}\left(\sum_{k=0}^{k_{\max}}\left(a^{k}\cos\left(2\pi b^{k}\left(x_{i}+0.5\right)\right)\right)\right)-n\sum_{k=0}^{k_{\max}}\left(a^{k}\cos\left(2\pi b^{k}\cdot 0.5\right)\right)$$

其中，$a=0.5$，$b=3$，$k_{\max}=20$。

搜索空间：$-1\leqslant x_i\leqslant 1$。

全局最优：$f(x)=0$，$x_i=0$。

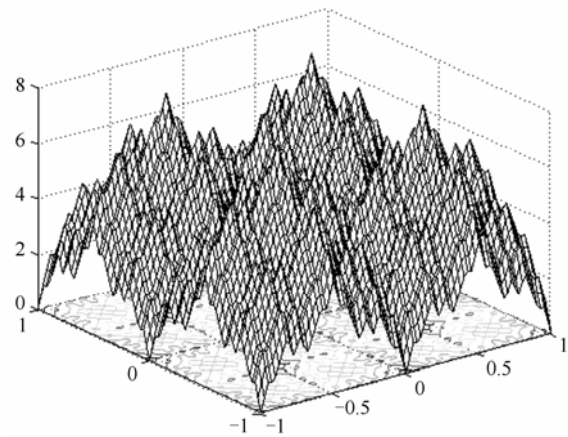

图 A.11　Weierstrass 函数的 3D 图形

11．Sum of Different Power函数

该函数为单峰测试函数，函数的 3D 图形如图 A.12 所示，表达式为

$$f(x)=\sum_{i=1}^{n}\left|x_i\right|^{i+1}$$

搜索空间：$-1\leqslant x_i\leqslant 1$。

全局最小值为：$f(x)=0$，$x_i=0$。

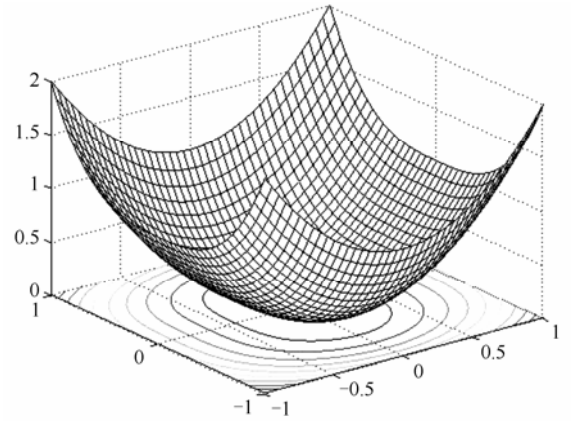

图 A.12　Sum of Different Power 函数的 3D 图形

12．Langermann函数

该函数为多峰测试函数，具有多个不均匀分布的局部极小值。函数的 3D 图形如图 A.13 所示，表达式为

$$f(x)=\sum_{i=1}^{m}c_i\,\mathrm{e}^{-\frac{1}{\pi}\sum_{j=1}^{n}(x_j-a_{ij})^2}\cdot\cos[\pi\sum_{j=1}^{n}(x_j-a_{ij})^2]$$

其中，c_i（$i=1,2,\cdots,m$）和a_{ij}（$i=1,2,\cdots,m$；$j=1,2,\cdots,n$）是常量，m 通常设置为 5。

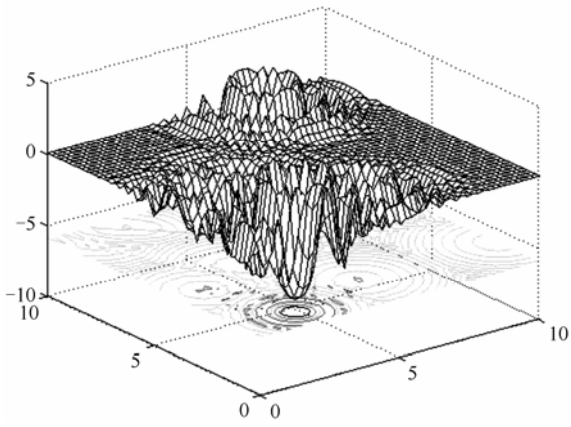

图 A.13　Langermann 函数的 3D 图形

函数的 2D 表达式为

$$f(x,y)=\sum_{i=1}^{m}c_i\,\mathrm{e}^{\frac{-(x-a_j)^2-(y-b_j)^2}{\pi}}\cdot\cos[\pi(x-a_i)^2+\pi(y-b_j)^2]$$

$$m=5,\quad a=[3,5,2,1,7],\quad b=[5,2,1,4,9],\quad c=[1,2,5,2,3]$$

13．Michalewicz函数

该函数为多峰测试函数，有 $n!$ 个局部极小值。参数 m 定义了峰谷和边缘的“陡峭度”。m 值较大时，会使搜索变得困难。函数的 3D 图形如图 A.14 所示，表达式为

$$f(x)=-\sum_{i=1}^{n}\sin(x_i)[\sin(\frac{ix_i^2}{\pi})]^{2m}$$

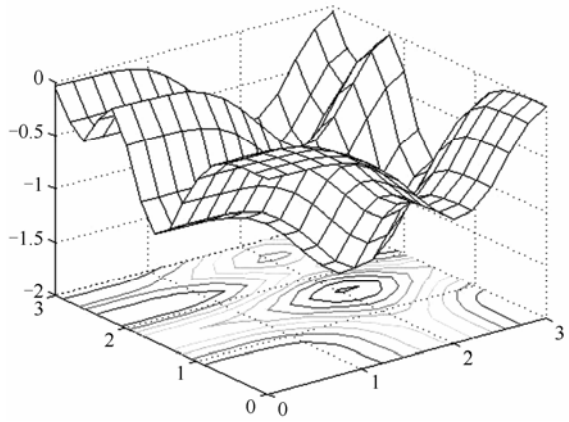

图 A.14　Michalewicz 函数的 3 维图形

通常设置 m=10。

搜索空间：$0\leqslant x_i\leqslant\pi$。

全局最优：$f(x)=-4.618$，n=5，以及 $f(x)=-9.66$，n=10。

函数的 2D 表达式为

$$f(x_1,x_2)=-\sin(x_1)\left[\sin(\frac{x_1^2}{\pi})\right]^{2m}-\sin(x_2)\left[\sin(\frac{2x_2^2}{\pi}\right]^{2m},\quad m=1$$

14．Branins函数

该函数为全局最优测试函数，仅有两个变量，三个相等的全局最优。函数的 3D 图形如图 A.15 所示，表达式为

$$f(x_1, x_2) = a(x_2 - bx_1^2 + cx_1 - d)^2 + e(1 - f)\cos(x_1) + e$$

其中，各常量的推荐值为：$a = 1$，$b = \dfrac{5.1}{4\pi^2}$，$c = \dfrac{5}{\pi}$，$d = 6$，$e = 10$，$f = \dfrac{1}{8\pi}$。

全局最优：$f(x_1, x_2) = 0.397887$，$(x_1, x_2) = (-\pi, 12.275), (\pi, 2.275), (9.42478, 2.475)$。

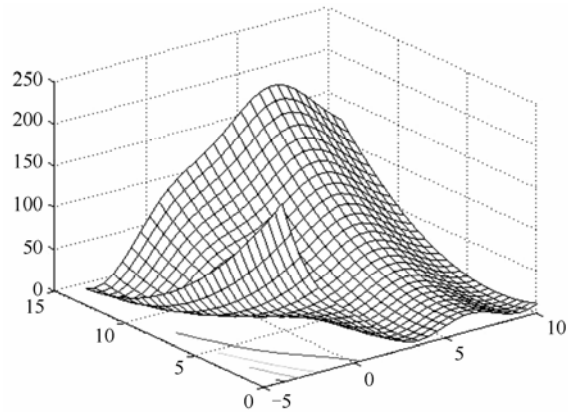

图 A.15　Branins 函数的 3D 图形

15．Easom函数

该函数为单峰测试函数，仅有两个变量，全局最优位于整个搜索空间中非常小的区域。函数的 3D 图形如图 A.16 所示，表达式为

$$f(x_1, x_2) = -\cos x_1 \cdot \cos x_2 \cdot \mathrm{e}^{-(x_1-\pi)^2-(x_2-\pi)^2}$$

搜索空间：$-100 \leqslant x_i \leqslant 100$。

全局最优：$f(x_1, x_2) = -1$，$(x_1, x_2) = (\pi, \pi)$。

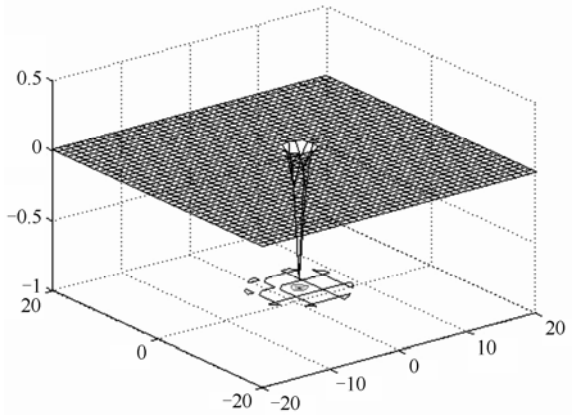

图 A.16　Easom 函数的 3D 图形

16．Goldstein-Price函数

该函数为全局最优测试函数，仅有两个变量。函数的 3D 图形如图 A.17 所示，表达式为

$$f(x_1,x_2)=[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2]\square$$
$$[30+(2x_1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2]$$

搜索空间：$-2\leqslant x_1$，$x_2\leqslant 2$。

全局最小为：$f(x_1,x_2)=3$，$(x_1,x_2)=(0,-1)$。

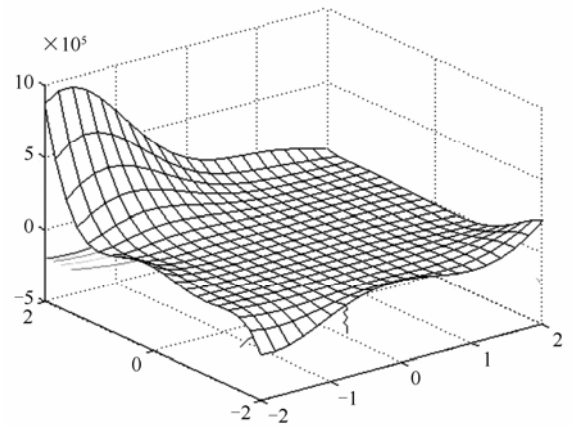

图 A.17　Goldstein-Price 函数的 3D 图形

17．Six-hump Camel Back函数

该函数为全局最优测试函数，仅有两个变量。在整个区域内有 6 个局部极小，其中的 2 个为全局最小值。函数的 3D 图形如图 A.18 所示，表达式为

$$f(x_1,x_2)=(4-2.1x_1^2+\frac{x_1^4}{3})x_1^2+x_1x_2+(-4+4x_2^2)x_2^2$$

搜索空间：$-3\leqslant x_i\leqslant 3$，$-2\leqslant x_i\leqslant 2$。

全局最小为：$f(x_1,x_2)=-1.0316$，$(x_1,x_2)=(-0.0898,0.7126)$ 或$(x_1,x_2)=(0.0898,-0.7126)$。

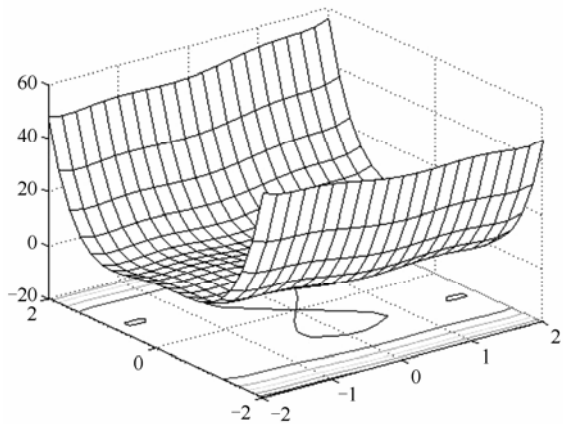

图 A.18　Six-hump Camel Back 函数的 3D 图形

18．De Jong第 5 函数

该函数为多峰测试函数，仅有 2 个变量。函数的 3D 图形如图 A.19 所示，表达式为

$$f(x_1,x_2)=\left\{0.002+\sum_{i=1}^{25}[j+(x_1-a_{1j})^6+(x_2-a_{2j})^6]^{-1}\right\}^{-1}$$

其中，

$$(a_{ij})=\begin{pmatrix}-32 & -16 & 0 & 16 & 32 & -32 & \cdots & 0 & 16 & 32\\ -32 & -32 & -32 & -32 & -32 & -16 & \cdots & 32 & 32 & 32\end{pmatrix}$$

函数也可以表示为

$$f(x_1,x_2)=\left\{0.002+\sum_{i=-2}^{2}\sum_{j=-2}^{2}[5(i+2)+j+3+(x_1-16j)^6+(x_2-16i)^6]^{-1}\right\}^{-1}$$

搜索空间：$-65.536\leqslant(x_1,x_2)\leqslant 65.536$。

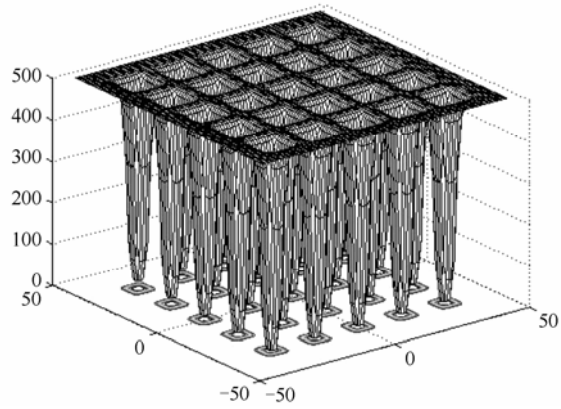

图 A.19　De Jong 第 5 函数

19．Drop Wave函数

该函数为多峰测试函数，仅有 2 个变量。函数的 3D 图形如图 A.20 所示，在$-2\leqslant(x_1,x_2)\leqslant 2$ 上的 3D 图形如图 A.21 所示，表达式为

$$f(x_1,x_2)=-\frac{1+\cos(12\sqrt{x_1^2+x_2^2})}{\frac{1}{2}(x_1^2+x_2^2)+2}$$

搜索空间：$-5.12\leqslant(x_1,x_2)\leqslant 5.12$。

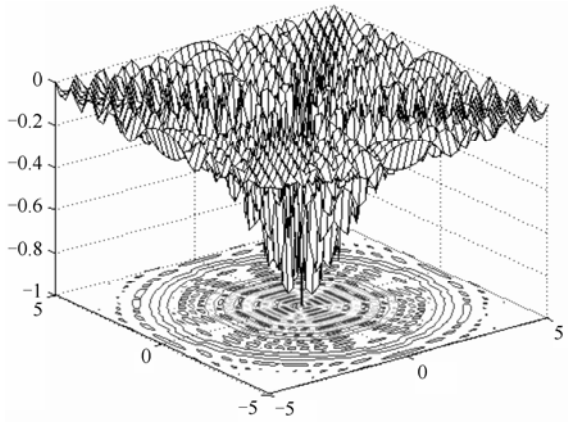

图 A.20　Drop Wave 函数的 3D 图形

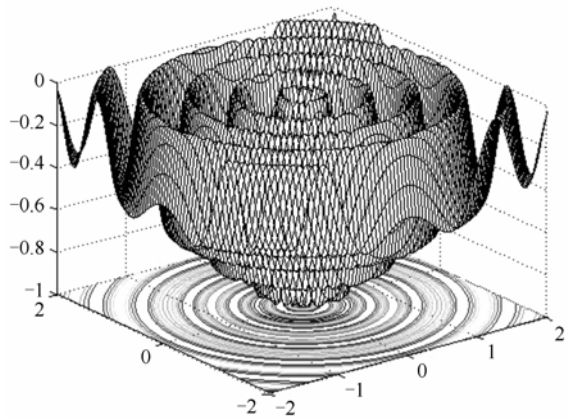

图 A.21　Drop Wave 函数在区域−2≤(x_1，x_2)≤2 上的 3D 图形

20．Shubert函数

该函数为多峰测试函数，仅有 2 个变量。函数的 3D 图形如图 A.22 所示，表达式为

$$f(x_1,x_2)=-\sum_{i=1}^{5} i\cos[(i+1)x_1+1]\sum_{i=1}^{5} i\cos[(i+1)x_2+1]$$

搜索空间：−5.12≤(x_1，x_2)≤5.12。

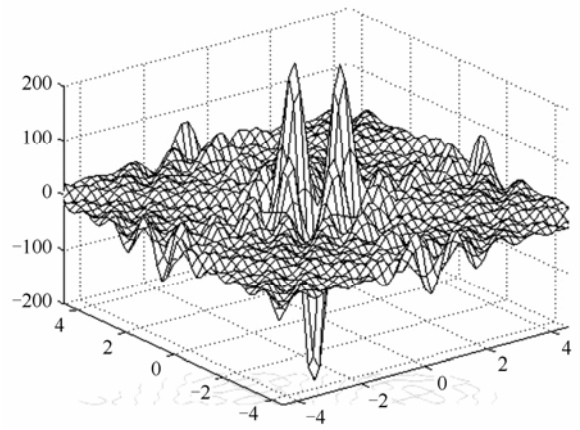

图 A.22　Shubert 函数的 3D 图形